

Bio-inspired Navigation Algorithm for GPS Denial Modes

By

Shriniwas Madhav Kolpuke

Submitted to the Department of Aerospace Engineering and the
Graduate Faculty of the University of Kansas
in partial fulfillment of the requirements for the degree of
Master of Science.

Committee members

Dr. Shawn Keshmiri, Chairperson

Dr. Craig McLaughlin

Dr. Ray Taghavi

Date defended: _____

The Thesis Committee for Shriniwas Madhav Kolpuke certifies

That this is the approved version of following thesis:

Bio-inspired Navigation Algorithm for GPS Denial Modes

Dr. Shawn Keshmiri, Chairperson

Date approved: _____

Abstract

The wide use of the Global Positioning System (GPS) for navigation has been persistent for a long time. However, in today's scenario when technologies are advancing the accuracy of positioning systems, there are various new threats and challenges emerging. The signal receivers for positioning systems are prone to spoofing. This external interference in the system is usually done by feeding false signal to the receiver. Though the dead reckoning method is still in use, any interference with GPS can still lead to disaster. Insects and birds are known to use solar position for guidance and it is widely accepted by researchers that some birds, such as pigeons, use solar position in their homing flight. There are similar studies performed on honeybees and monarch butterflies. The use of solar position by these insects and birds brings up the question of whether a mathematical model can be used to replicate the results for real-life navigation and can a bio-inspired navigation algorithm like this be implemented. Solar position algorithms are already in wide use. The solar position algorithms available calculate the azimuth and zenith/incidence angles for the solar position at any given point of time when the position of the observer is known. The objective for navigation is to find an observer's position from solar position to present an alternative to GPS for navigational use. This document proposes the method for calculating the observer's position when the azimuth and zenith/incidence angles for the solar position, attitude of aircraft and the accurate time are known. The approach proposed is that the position of the observer is to be calculated by adopting the concept of reversing the ENEA algorithm where instead of calculating the solar position using an observer's position, one will be calculating the position of observer from the solar position and time.

Acknowledgements

I would like to start this by thanking one person especially without whom this project might not have been possible and that is **Dr. Shawn Keshmiri**. His dedication towards the advancement of his students is unparalleled. Without his support, understanding, and patience I wouldn't be standing in this position today. I couldn't have chosen a better adviser towards my Master's degree. Words won't be enough to express how grateful I am to you. Thank you, Dr. Keshmiri for believing in me. **Dr. Gonzalo Garcia**, I have immensely enjoyed our discussions of aircraft motion and dynamics. It was insightful and helped me a lot in understanding the complexity of aircraft dynamics.

I want to thank, **Dr. Craig McLaughlin** for his support during my studies. I thoroughly enjoyed your coursework of Spacecraft Systems and Statistical Orbit Determination, it motivated me towards the selection of this topic. **Dr. Ray Taghavi**, I cannot describe how much fun it was to learn Rocket Propulsion with you. Thank you so much for the opportunity. I am grateful to **Dr. Haiyang Chao**, the GPS study I did with you formed the foundation for my studies. I am thankful to **Prof. Erik Van Vleck** and **Xuemin Tu** from Department of Mathematics for helping me get a deep understanding of Numerical Methods.

Family and Friends, I cannot express how thankful I am to you for supporting my decision of leaving my field of expertise and going for something new. It wasn't easy but I made it.

Contents

| | |
|--|-----------|
| 1 Introduction | 1 |
| 2 Observer's position calculation model | 4 |
| 2.1 Algorithm derivation..... | 4 |
| 2.2 Algorithm flowchart..... | 8 |
| 3 Sensors | 9 |
| 4 Algorithms to solve the equations | 12 |
| 4.1 Newton's method | 12 |
| 4.2 Trust region method | 13 |
| 5 Results and discussion | 14 |
| 5.1 Newton's method | 15 |
| 5.2 Trust region method..... | 16 |
| 5.2.1 Output of trust region algorithm..... | 18 |
| 5.2.2 Correlation of the error..... | 22 |
| 5.2.2.1 Error caused by time of day..... | 23 |
| 5.2.2.2 Error caused by Day..... | 23 |
| 5.2.2.3 Error correlation with aircraft heading and attitude..... | 24 |
| 5.2.3 Investigation of sensitivity of error caused by sensors..... | 27 |
| 5.2.4 Miscellaneous..... | 32 |
| 5.2.4.1 Behavior of algorithm..... | 32 |
| 5.2.4.2 Execution time and data rate of algorithm..... | 33 |
| 5.2.4.3 Accuracy of GPS at stationary position..... | 34 |
| 6 Future work | 37 |
| 7 Conclusion | 40 |
| 8 References | 41 |

| | |
|---|-----------|
| 9 Appendix | 43 |
| A. Output of Trust region algorithm..... | 43 |
| B. Correlation of error with time of day and day..... | 73 |
| C. Sensitivity of error caused by sensors..... | 76 |
| D. MATLAB code for ENEA algorithm..... | 86 |
| E. MATLAB code for SPA algorithm..... | 90 |
| F. MATLAB code for Michalsky algorithm..... | 100 |

Abbreviations

α – Geocentric right ascension

α_t - Topo-centric right ascension

Date – (D – Day, M – Month and Y - Years)

Δ - Difference between UT (UTC) and TT (* $\Delta = TT - UT$)

δ – Declination angle

δ_t - Topo-centric declination

$\Delta\alpha$ – Parallax correction to right ascension

Δe – Atmospheric refraction correction due to solar elevation

$\Delta\gamma$ – Correction to geocentric longitude due to nutation

e_0 - Solar elevation angle due to refraction correction

γ – Geocentric solar longitude

Γ – Local topo-centric sun coordinates: Azimuth

h - Local hour angle of sun

h_t - Topo-centric hour angle

INT – Function which rounds its arguments to the nearest integer towards.

L – Longitude

M – Month of UTC

ε – Earth axis inclination

φ - Observer's Latitude

t_G - Julian Day

t - Julian Ephemeris day

θ - Observer's Longitude

UT – Fractional UT of the day measured in the hours from the Greenwich midnight and the minutes and seconds must be converted into fraction of hours.

z – Local topo-centric sun coordinates: Zenith

List of Figures

| | |
|---|----|
| 2.1 Observer's position calculation algorithm flowchart..... | 8 |
| 3.1 nanoSSOCD60 | 9 |
| 3.2 Sun sensor mount structure..... | 10 |
| 3.3 CurvACE sensor..... | 11 |
| 5.1 GPS position..... | 14 |
| 5.2 True and estimated position..... | 16 |
| 5.3 DG808 unmanned aircraft..... | 18 |
| 5.4 True and estimated position | 19 |
| 5.5 True and estimated position comparison..... | 20 |
| 5.6 Position difference with respect to time..... | 21 |
| 5.7 Correlation of error with time of day..... | 23 |
| 5.8 Correlation of error with day..... | 24 |
| 5.9 True position, estimated position and comparison..... | 25 |
| 5.10 Position difference, altitude and true velocity..... | 25 |
| 5.11 Position difference, latitude and longitude comparison..... | 26 |
| 5.12 Correlation of error..... | 26 |
| 5.13 Correlation of error..... | 27 |
| 5.14 Error variations with respect to inaccuracies in sensor data..... | 28 |
| 5.15 Error variations with respect to inaccuracies in Azimuth..... | 29 |
| 5.16 Error variations with respect to inaccuracies in Zenith..... | 29 |
| 5.17 Error variations with respect to inaccuracies in Azimuth and Zenith..... | 30 |

| | |
|--|----|
| 5.18 Estimated and true position..... | 30 |
| 5.19 Estimated and true position..... | 31 |
| 5.20 True position, estimated position and comparison..... | 32 |
| 5.21 True position, estimated position and comparison..... | 33 |
| 5.22 The location and setup..... | 34 |
| 5.23 u-blox on NE frame accuracy..... | 36 |
| 5.24 NovAtel on NE frame accuracy..... | 36 |

List of Tables

| | |
|---|----|
| 3.1 Sun sensors comparison..... | 10 |
| 5.1 Comparison and average of error in North and East standard deviation..... | 22 |
| 5.2 Execution time and data rate of algorithm..... | 33 |
| 5.3 Performance comparison of u-blox LEA6H and NovAtel OEM615..... | 35 |

Chapter 1

Introduction

In the navigation field, various sensors or combinations of sensors have been used for position calibration. The use of GPS and the dead reckoning method have been a common practice. However, in today's scenario the reliability of the GPS system cannot be trusted when it is prone to spoofing. The way this interference happens is by feeding false signals on the inputs on the same frequency as that of GPS. However, even though the false frequency is known to be stronger than that of original GPS frequency, which makes it easier to differentiate between them, the question of reliability remains unanswered as the modern electronic warfare devices are achieving advances. The use of celestial objects for this purpose is a possible solution. The use of celestial objects for navigation is not new. The use of a solar compass and the North Star (along with Ursa Minor and Ursa Major) by sailors is well known and also recent studies show the use of the solar position by many species for navigation.

Insects and birds are known to use the sun's position for guidance. Researchers widely accept that some birds, including pigeons, use the sun's position in their homing flight. However, in this case the sun does not tell the animal its position and which direction it should fly in order to get home. The bird first calibrates its direction using other sources of information and after that they follow the appropriate course using the sun. The study by researchers has shown that birds use the geomagnetic field to determine compass directions, but inputs from the sun compass are preferred when the sun is visible. [3]

The study performed by Menzel and Greggers [4] on the memory structure of navigation in honeybees outlines the use of ground structure in relation to the sun compass used by honeybees to get back to their hives or to reach previously explored forage. In catch-and-release experiments they found that the honeybees need to learn, identify, distinguish and localize the objects in the environment for navigation. In their study they have adopted the view that navigational memory in honeybees can be best conceptualized by a cognitive map that stores the spatial relation between multiple landscape features they use for navigation. [4]

This same concept of the use of atmospherically scattered sunlight to reliably determine the absolute heading by many insects has been implemented by Ashkanazy and Humbert [5] to get the absolute heading as an alternative to the dead reckoning methods and the magnetic compass. The physical model they implemented uses the visual inputs from the structure based on Dorsal Rim Area (DRA) along with the temporal circadian rhythm and the polarization pathway where

the combination of detected relative bearing from the sun and predicted sun azimuth is used to calculate absolute heading. [5]

In the studies of migrating Eastern North American monarch butterflies performed by Shlizerman [11], they present a model that integrates neuronal oscillations of the sun's horizontal position and the circadian clock to direct flight and their model explains flight simulator tracks and proposes a space-time integration mechanism for directional flight. The monarchs use a circadian clock housed in the antennae which is capable of keeping track of time-of-day and to control the sun compass timings. The solar azimuth is detected by the monarch's eyes which are shaped as half spheres separated by 180° from each other and are composed of many individual lenses. The results of their study demonstrates a simple neuronal integration mechanism based on matching spiking rates of clock and azimuth neural signals to provide angular position control and this mechanism helps monarch butterflies to make their journey and reach their overwintering sites relying on limited food sources on their flight path. [11]

The basic idea of this project is to calibrate the observer's position when the azimuth and zenith/incidence angles for the sun's position are known. The solar position algorithms available calculate the azimuth and zenith/incidence angles for the sun's position at any given point of time when the position of the observer is known. Here the goal is to achieve exactly the opposite to present an alternative to the GPS sensor for navigational use. In this study the main objective is to achieve the exact location of the observer in real time with very accurate time, known attitude of aircraft and solar position as inputs. The calculation of solar position in terms of zenith and azimuth is to be performed by sensors. The traditional method for air navigation uses a combination of rate gyros, accelerometers, and pressure sensors. The work presented in this study will add on to this method to make it more reliable.

There have been plenty of algorithms proposed over the years for the calculation of solar position with respect to the observer's position and most of them have been used for solar radiation applications. For this study existing algorithms are selected based on the accuracy and the effective time duration for that specific algorithm proposed by the respective author.

Three algorithms found to be relevant from the proposed time duration are the Michalsky algorithm [6], SPA algorithm [2], and ENEA algorithm [1]. However, based on the accuracy requirements for this project only the SPA and ENEA algorithms are relevant with estimated accuracy of 0.0003° and 0.0027° respectively. The effective time duration of these algorithms depends upon the reference day for the Julian time used in the algorithm by these specific authors. In case of the SPA algorithm the uncertainties equal to $\pm 0.0003^\circ$ for the years -2000 to 6000 as it uses the reference day of January 1, in year -4712 at 12:00:00 UT [1] which is the actual start day of the Julian calendar. However, the Michalsky algorithm it is limited to the period of 1950 to 2050 with uncertainty greater than $\pm 0.01^\circ$ because it uses the reference day of noon January 1, 2000 [6] [2]. The ENEA algorithm has the maximal uncertainty of $\pm 0.0027^\circ$ for the time period of 2003 to 2022 and it uses noon January 1, 2003 as its reference day for Julian date [1].

The basic concept behind the calculations in all of these algorithms is first to calculate the position of the planet, earth in this case, with respect to the sun at any given point of time in terms of heliocentric longitude and latitude along with the radius vector, and then achieve the values of geocentric longitude and latitude and finally calculate the topo-centric coordinates of the observer in terms of azimuth and zenith angles. These algorithms are usually designed around the periodic terms of the respective planet explained in Chapter 31 and Appendix II of the book ‘Astronomical Algorithms’ [7].

Various accuracy improvement parameters have been added in between to add the effects of different aspects affecting the planetary calculations like nutation in longitude and obliquity, aberration correction along with right ascension and declination angles. These algorithms also have parameters like refraction correction to address errors caused by atmospheric effects. In case of the ENEA algorithm, the polynomial correction has also been used while calculating the heliocentric longitude of earth.

All three algorithms have been studied and implemented to verify the accuracy in real time. After a thorough analysis, it was found that SPA algorithm uses logical statements in multiple parts of algorithm and the reverse approach cannot be implemented. The ENEA algorithm was selected to apply the reverse approach. At this point the main objective is to achieve the 2D position of the observer as the third parameter, elevation can be easily obtained with the use of a Pitot tube, which is believed to be very accurate. Apart from SPA algorithm, no other algorithm has elevation parameter as a part of it. The process of obtaining the observer’s position from the known time, attitude of aircraft and solar location is presented in the next chapter.

Chapter 2

Observer's Position Calculation Model

The observer's position calculation model is based on the selected solar position calculation algorithm's accuracy. After a careful comparative study as mentioned in Chapter 1, the ENEA¹ algorithm has been selected to implement the idea of a reverse approach to calculate the position of the observer. The assumption is that the solar position, attitude of aircraft and the accurate time are known. The objective is to find the position of the observer in terms of longitude and latitude. The process of finding the equations to calculate longitude and latitude is as follows.

2.1 Algorithm derivation

The ENEA algorithm has zenith and azimuth formula as follows.

$$\text{Zenith: } z = \frac{\pi}{2} - e_0 - \Delta e \quad (\text{I}) \quad (23^{[1]})$$

$$\text{Azimuth: } \Gamma = \text{atan2}(sh_t, ch_t * \sin\varphi - \tan\delta_t * \cos\varphi) \quad (\text{II}) \quad (24^{[1]})$$

Where:

e_0 - Solar elevation angle without refraction correction

Δe - Atmospheric refraction correction due to solar elevation

h_t - Topo-centric hour angle

sh_t - *approximate cosine of h_t*

ch_t - *approximate sine of h_t*

φ - Observer's latitude

δ_t - Topo-centric declination

¹ The selected algorithm is the work of Roberto Grena, in the paper 'An algorithm for the computation of the solar position' and generally known as ENEA (Centro Ricerche Casaccia, via Anguillarese 301, Roma, Italy) algorithm.

The time scales t_G and t used in this algorithm are the Julian Day and the Julian Ephemeris day respectively. They have been shifted to start at noon, 1 January 2003. [1]

$$t_G = INT(365.25(Y - 2000)) + INT(30.6001(M + 1)) + D + \frac{UT}{24} - 1158.5 \quad (2^{[1]})$$

$$t = t_G + \frac{\Delta}{86400}$$

M – Month of UTC. (*If M is 1 or 2, M must be increased by 12 and Y must be reduced by 1) [1]

INT – Function which rounds its arguments to the nearest integer towards 0 (*i.e. $INT(7.8) = 7$, $INT(-6.6) = -6$) [1]

UT – Fractional UT of the day measured in the hours from the Greenwich midnight and the minutes and seconds must be converted into fraction of hours. [1]

$Date$ – (D – Day, M – Month and Y - Years) [1]

Δ - Difference between UT (UTC) and TT (* $\Delta = TT - UT$) [1]

Geocentric longitude due to nutation:

$$\Delta\gamma = (8.33e - 5) * \sin((9.252e - 4) * t - 1.173) \quad (8^{[1]})$$

In the ENEA algorithm, until the calculation of the local hour angle of the sun, knowledge of the observer's position is not needed as they are only time based functions. The proposed algorithm needs these time dependent parameters to calculate the observer's position. After the calculation of the time dependent parameters, the algorithm will use the derived equations III and IV for the calculation of the observer's position.

Local hour angle of the sun:

$$h = 6.30038809903 * t_G + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha \quad (14^{[1]})$$

Parallax correction to right ascension:

$$\Delta\alpha = (-4.26e - 5) * \cos\varphi * \sinh \quad (15^{[1]})$$

Topo-centric right ascension:

$$\alpha_t = \alpha + \Delta\alpha \quad (16^{[1]})$$

Topo-centric declination:

$$\delta_t = \delta - (4.26e - 5) * (\sin\varphi - \delta\cos\varphi) \quad (17^{[1]})$$

Topo-centric hour angle:

$$h_t = h - \Delta\alpha \quad (18^{[1]})$$

$$ch_t = \cosh + \Delta\alpha * \sin h \text{ (approximate cosine of } h_t) \quad (19^{[1]})$$

$$sh_t = \sinh - \Delta\alpha * \cos h \text{ (approximate sine of } h_t) \quad (20^{[1]})$$

Solar elevation angle without refraction correction:

$$e_0 = \arcsin(\sin\varphi * \sin\delta_t + \cos\varphi * \cos\delta_t * ch_t) \quad (21^{[1]})$$

Atmospheric refraction correction due to solar elevation:

$$\Delta e = \frac{0.084217 * P}{[(273 + T) * \tan(e_0 + \frac{0.0031376}{(e_0 + 0.089186)})]} \quad (22^{[1]})$$

The equations 14-22 from ENEA algorithm which contains the components of the observer's location, are substituted in the zenith and azimuth equations to achieve the equation which has longitude and latitude as variables. The remaining parameters in the equations are known, and can be used to calculate all time based parameters. The derived equations are as follows,

Zenith:

$$z = \frac{\pi}{2} - (\text{asin}(\sin\varphi * \sin(\delta - 4.26e - 5 * (\sin\varphi - \delta\cos\varphi)) + \cos\varphi * \cos(\delta - 4.26e - 5 * (\sin\varphi - \delta\cos\varphi)) * (\cos(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha) +$$

$$- \alpha) + ((-4.26e - 5 * \cos\varphi * \sin(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha))) * \sin(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha))) + \frac{1}{(273+\Gamma)} *$$

$$\frac{0.084217 * P}{(\text{asin}(\sin\varphi * \sin(\delta - 4.26e - 5 * (\sin\varphi - \delta\cos\varphi)) + \cos\varphi * \cos(\delta - 4.26e - 5 * (\sin\varphi - \delta\cos\varphi)) * (\cos(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha) + \tan(((-4.26e - 5 * \cos\varphi * \sin(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha))) * \sin(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha))) + 0.0031376) * \sin(6.30038809903 * \sin(\delta - 4.26e - 5 * (\sin\varphi - \delta\cos\varphi)) + \cos\varphi * \cos(\delta - 4.26e - 5 * (\sin\varphi - \delta\cos\varphi)) * (\cos(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha) + (-4.26e - 5 * \cos\varphi * \sin(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha))) * \sin(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha))) + 0.089186$$

7

(III)

Azimuth:

$$\Gamma = \text{atan2}((\sin(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha) - (-4.26e - 5 * \cos\varphi * \sin(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha)) * \cos(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha)), (\cos(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha) + (-4.26e - 5 * \cos\varphi * \sin(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha)) * \sin(6.30038809903 * t_c + 4.8824623 + 0.9174 * \Delta\gamma + \theta - \alpha)) * \sin\varphi - \tan(\delta - 4.26e - 5 * (\sin\varphi - \delta\cos\varphi)) * \cos\varphi)$$

(IV)

The only unknown parameters in equation (III) and equation (IV) will be the observer's longitude (θ) and observer's latitude (φ). These two nonlinear equations can be further solved to get the position of observer.

2.2 Algorithm flowchart

The flow for the calculation of the observer's position is shown in Figure 2.1. The calculations are presented in six sections. The first section calculates the time parameters and addresses the necessary time conversions. The heliocentric parameters are calculated in second section. The third, fourth and fifth section calculates correction in geocentric longitude, earth axis inclination and geocentric global solar coordinates respectively. The final section uses the equations III and IV presented in this algorithm to calculate the observer's position.

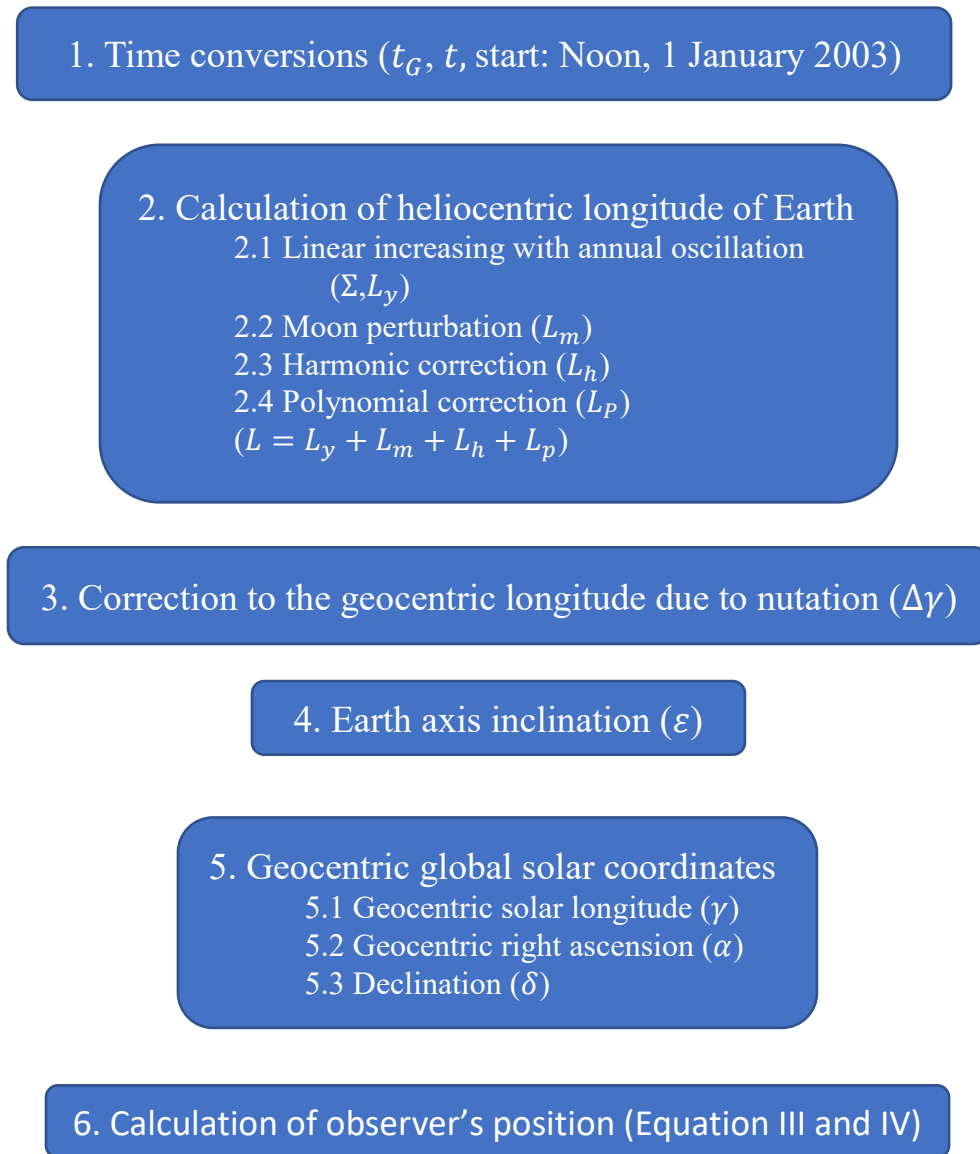


Figure 2.1. Observer's position calculation algorithm flowchart

Chapter 3

Sensors

The availability of sensors is no longer an assumption. Sensors² to calculate both azimuth and zenith are available. The sensors needed to calculate zenith are widely used in the aerospace industry and are known as sun sensors. The name of the sensor needed to calculate the azimuth is CurvACE.



Figure 3.1. nanoSSOCD60 [13]

The sun sensors have a wide variety of applications. They are used on satellites for attitude determination, satellite solar panel positioning, solar position determination, balloons and UAVs control and more [12] [13] [14] [15]. The sun sensor will be used to determine the exact zenith of the sun. The comparison of the sun sensors from Solar MEMS technologies are given in the following table.

² This chapter is included to show the possibility of calculating the intensity and position of the sun. The study conducted is strictly limited to the mathematical validation of the concept and the sensors are not used in this study.

| | nanoSSOC-A60 [12] | nanoSSOC-D60 [13] | SSoC-A60 [14] | SSoC-D60 [15] |
|--------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| Type | 2 orthogonal axes | 2 orthogonal axes | 2 orthogonal axes | 2 orthogonal axes |
| Field of View | $\pm 60^\circ$ | $\pm 60^\circ$ | $\pm 60^\circ$ | $\pm 60^\circ$ |
| Accuracy | $< 0.5^\circ$ | $< 0.5^\circ$ | $< 0.3^\circ$ | $< 0.3^\circ$ |
| Precision | $< 0.1^\circ$ | $< 0.1^\circ$ | $< 0.05^\circ$ | $< 0.05^\circ$ |
| Power supply | 3.3V / 5V | 3.3V / 5V | 5-12V | 5V |
| Power consumption | < 2 mA | < 23 mA | 12-36mW | 70 mA |
| Size | 27.4 x 14 x 5.9 mm | 43 x 14 x 5.9 mm | 40 x 30 x 12 mm | 60 x 30 x 12 mm |
| Weight | 4 g | 6.5 g | 25 g | 35 g |
| Temperature range | -30 to +85 $^\circ\text{C}$ | -30 to +85 $^\circ\text{C}$ | -40 to +85 $^\circ\text{C}$ | -40 to +85 $^\circ\text{C}$ |

Table 3.1. Sun sensors comparison

The field of view of these sensors is limited to $\pm 60^\circ$ which will not be able to cover the full spectrum for our application. To resolve this issue, five sensors can be used with the mounting shown in Figure 3.2.

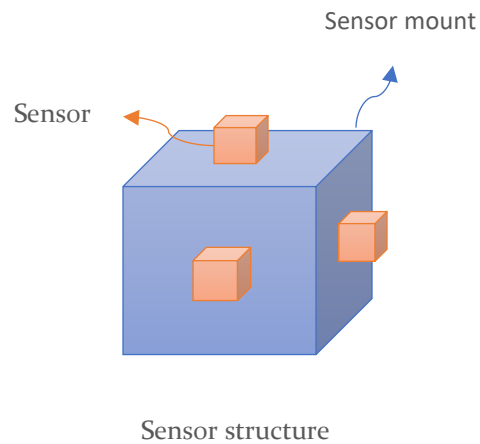


Figure 3.2. Sun sensor mount structure

Five sensors are mounted as one each on sides of cube except the bottom to cover the full field of view to calculate an accurate solar zenith.

A majority of animal species including migrating Eastern North American monarch butterflies has the vision mediated by compound eyes [11] [16]. Floreano [16] have succeeded in fabricating a prototype of a miniature curved artificial compound eye named CurvACE which can calculate the solar azimuth based on photodetectors. The field of view of the CurvACE is $180^\circ \times 60^\circ$ [16]. The CurvACE prototype has a bent rectangular array of 42 columns of 15 artificial ommatidia, totaling 630 artificial ommatidia. The interommatidial angle and acceptance angle are both about $\sim 4.2^\circ$. [16]

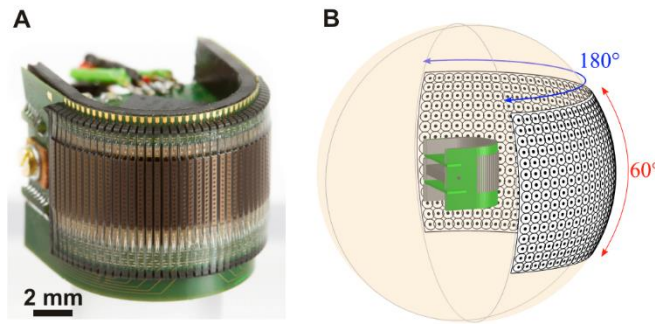


Figure 3.3. CurvACE sensor A) CurvACE sensor B) FOV (Field of view) of CurvACE sensor [16]

Two of these sensors can be used to cover the full 360° spectrum and the solar azimuth can be calculated with respect to north. Although the current resolution and accuracy of this curved artificial compound eye sensor may not be sufficient for our purpose, this is the exact kind of sensor that is needed to find the azimuth.

Sensor module integration with body frame:

The sensors will be mounted on the aircraft or a moving vehicle but the sensors always need to be parallel to the X-Y plane to get accurate readings for the zenith and azimuth angles. This can be solved by a rotation matrix which will always convert the reading of zenith and azimuth angles from the body frame to the frame parallel to the X-Y plane. The use of cosine and sine rules can resolve this issue.

Chapter 4

Algorithms to Solve the Equations

Two different approaches have been taken to solve the two non-linear equations to get the position of the observer. 1) Newton's numerical method and 2) the trust region method.

4.1 Using Newton's Method

In Newton's method the formula for the calculation of the next value from the previous or last known value is usually given as follows for the multi-dimensional case [9],

$$x_{k+1} = x_k - J_x^{-1} * f(x_k) \quad (\text{V})$$

Where the approximation x_{k+1} to a root of $f(x) = 0$ is computed from the approximation x_k using the above equation.

Newton's method can be used to solve two nonlinear equations with two unknown variables, in this case θ and φ . The functional equations can easily be achieved by taking all terms to one side and setting the equation equal to zero. The two main equations can be referred as $f_1(\theta, \varphi)$ and $f_2(\theta, \varphi)$ and the main function can be written as follows for this multi-dimensional case.

$$f(x) = \begin{bmatrix} f_1(\theta, \varphi) \\ f_2(\theta, \varphi) \end{bmatrix} \quad (\text{VI})$$

The derivative term or the Jacobian term in this case can be represented as follows,

$$f'(x) = J_x = \begin{bmatrix} \frac{\partial f_1(\theta, \varphi)}{\partial \theta} & \frac{\partial f_1(\theta, \varphi)}{\partial \varphi} \\ \frac{\partial f_2(\theta, \varphi)}{\partial \theta} & \frac{\partial f_2(\theta, \varphi)}{\partial \varphi} \end{bmatrix} \quad (\text{VII})$$

From equations (V), (VI) and (VII) the next term (prediction) can be calculated. The calculated term can again be used as input in the next loop to calculate the next term and so on.

4.2 Using Trust Region Method

The trust region method has matured over more than five decades now. Iterative method for optimization are usually divided into two categories as line search methods and trust region methods. The line search optimization algorithms obtain a search direction at each iteration and search along that direction to obtain a better estimate. The trust region approach is associated with approximation. For an optimization problem if we have a current guess of solutions, an approximate model can be constructed near that point and the solution of that approximate model can be taken as the next iteration point. In the trust region algorithm, the approximate model can only be trusted in the region near the current iteration. The region in which the approximate model is trusted is called the trust region. [17]

The MATLAB software has been used in this project exclusively and it has inbuilt function ‘fsolve’ which is based on trust region method to solve nonlinear equations. This inbuilt algorithm solves a set of n nonlinear functions $F_i(x)$ where n is number of components of vector x with aim of finding a vector x that makes all $F_i(x)=0$ [18]. This function consists of three algorithms as follows [18],

- Trust region-reflective
- Trust region dogleg
- Levenberg-Marquardt

The result of these methods are discussed in next chapter.

Chapter 5

Results and Discussion

The GPS receiver data from actual flight tests of DG808 unmanned aircraft has been used to test the algorithm. This way it becomes easy to evaluate the accuracy of the position output of the algorithm in comparison with GPS positions. This GPS position data has also been used to calculate zenith and azimuth angles of sun's position by using the ENEA algorithm developed during this study. Now with the availability of GPS time along with respective zenith and azimuth angles, the position of the observer can be calculated by using this algorithm. The first position value from the GPS in terms of longitude and latitude is to be used as last known value.

The path followed by the GPS position has been used as an example for this analysis and it has been shown as follows in Figure 5.1(a) and 5.1(b).

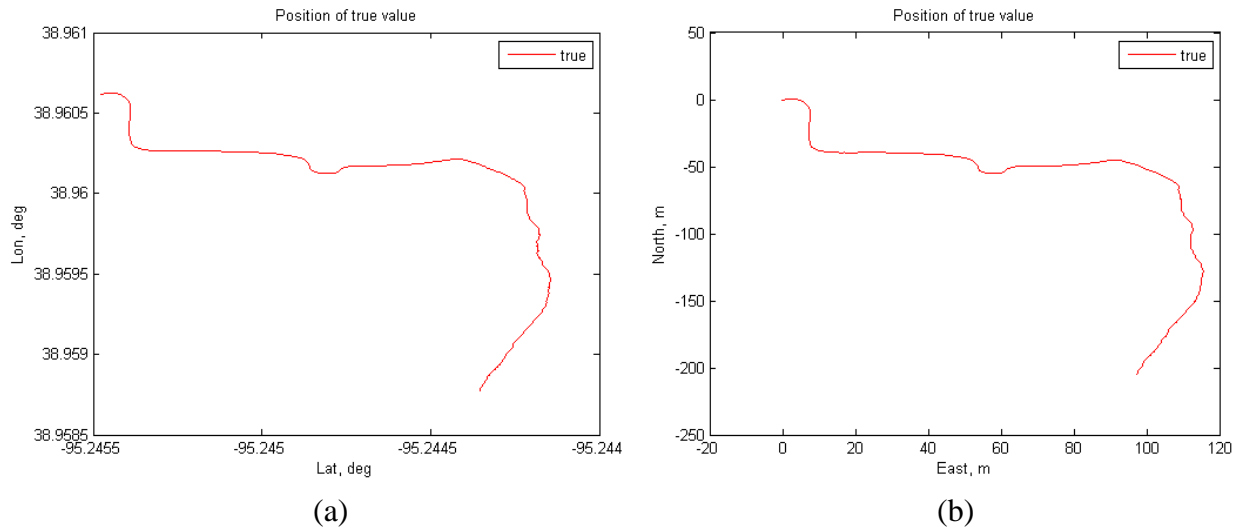


Figure 5.1. GPS position, (a) GPS 2D positions in LLA (Geodetic) coordinate system, (b) GPS 2D positions in NED (Local) coordinate system.

These GPS readings are from u-blox GPS receiver which operates on 4Hz frequency (Time step: 250 milliseconds).

5.1 Newton's Method

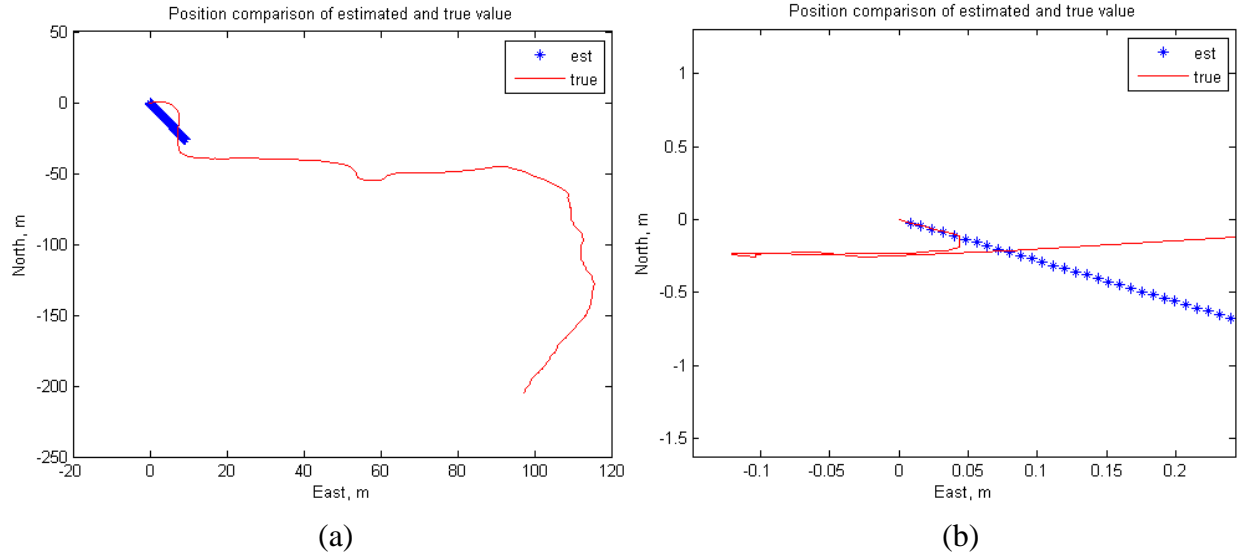
The solution of the algorithm using Newton's method has few deficiencies as the Jacobian is badly scaled for which the presence of many trigonometric and inverse terms can be a probable reason. The badly scaled Jacobian usually gives a big value as output which results into the mechanism not detecting the output and overlooking it at each iteration. Generally, in these cases, the Jacobian acts as a time step so the mechanism can detect the correct value. In this scenario, it is common to multiply a time step to the Jacobian in Newton's method. The equation (V) therefore can be redefined as follows,

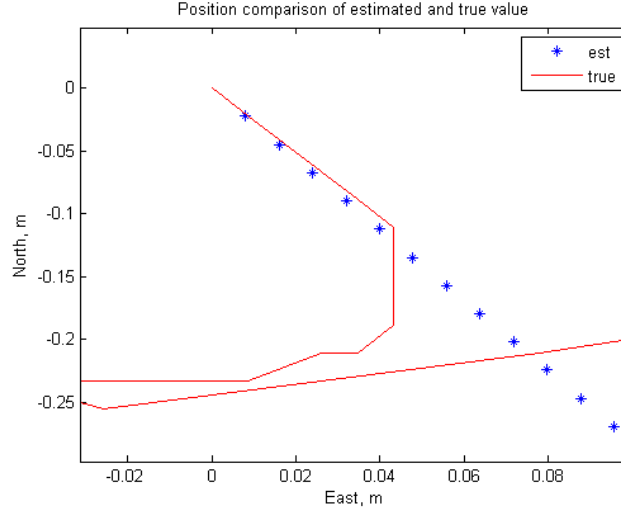
$$x_{k+1} = x_k - T_s * (J_x^{-1} * f(x_k)) \quad (\text{XIII})$$

Where: T_s – Time step

$$d_k = - (J_x^{-1} * f(x_k))$$

At this point, the solution can be achieved by controlling the number of iterations, maximum error, and time step. The output for the single point calculation gives the position accurately within few centimeters of accuracy from the original position. However, this pattern is not followed in the dynamic mode when the complete path has been applied as an input and the algorithm has to predict next position output at each point of time. This behavior has been shown as follows in Figure 5.2 (a), 5.2 (b) and 5.2 (c).





(c)

Figure 5.2. True and estimated position (a) True and estimated position in NED (Local) coordinate system, (b) and (c) True and estimated position at the beginning of the path in NED (Local) coordinate system

The probable cause found for this behavior so far indicates the behavior is due to the time step which has been added to help with convergence and accuracy. Any reduction in time step also reduces the accuracy and with time step close to 1, it does not converge. The inability of Newton's method to solve these equations can be also attributed to the badly scaled / ill-conditioned Jacobian matrix. These equations consist plenty of inverse trigonometric terms and any small change in input values can change the output drastically. The differentiation of the equations to get Jacobian matrix has been performed with the help of symbolic toolbox in MATLAB for accuracy but there is still a possibility that the Jacobian is badly scaled due to the complexity of equations. The model can work in dynamic mode only when the time step is standardized which will depend on the velocity and direction. This makes it impossible to make this model work in real time scenario as having the knowledge of velocity and direction in advance is not possible.

One more cause of concern in this method is the processing power needed to execute this operation as the mentioned accuracy has been achieved with the 10000 iterations, $1e-15$ error tolerance, and $1.3469e-13$ as a time step. This creates the necessity of a method which can not only solve these highly non-linear equations but also doesn't need heavy processing power. That way the general hardware available on aircraft can process the calculations to get aircraft's position.

5.2 Trust Region Method

The inbuilt trust region method in MATLAB has used trust region dogleg algorithm to solve equations (III) and (IV). In Newton's method, if $J(x_k)$ is singular, Newton's step d_k is not defined and may be expensive to compute. Newton's method may not converge if the starting point

is far from the solution. The use of trust region techniques improve robustness when starting far from the solution and handles the case if Jacobian matrix is singular. To use this, a merit function is needed to decide if x_{k+1} is better or worse than x_k . [18]

$$\min_d f(d) = \frac{1}{2} F(x_k + d)^T F(x_k + d) \quad (\text{IX})^{[18]}$$

The Newton step d_k is root of $M(x_k + d) = F(x_k) + J(x_k)d$ and so it is also minimum of $m(d)$,

$$\min_d m(d) = \frac{1}{2} F(x_k)^T F(x_k) + d^T J(x_k)^T F(x_k) + \frac{1}{2} d^T J(x_k)^T J(x_k) d \quad (\text{X})^{[18]}$$

Thus $m(d)$ is better choice of merit function $f(d)$, then the trust-region subproblem is,

$$\min_d [\frac{1}{2} F(x_k)^T F(x_k) + d^T J(x_k)^T F(x_k) + \frac{1}{2} d^T J(x_k)^T J(x_k) d] \quad (\text{XI})^{[18]}$$

Such that $\|Dd\| \leq \Delta$. This subproblem can be effectively solved using a dogleg strategy. [18]

The 'fsolve' function, when used for a set of nonlinear equation is used in a format given in the following template. The variation in output is supposed to be minimum as the frequency of the input data is 20Hz which means the variation in the time (difference of time in each interval) is 50ms. The maximum speed of the experimental aircraft DG808 is 35Knots which is approximately 18m/s. With given speed of data, the maximum distance traveled by the vehicle is approximately 0.9m per iteration. This distance traveled will cause a very small change in the values of longitude and latitude.

```
function F = Sol_trust_region(x)
format long
% Input Values-----
tg = __; %Julian Day
t = __; %Julian Ephemeris day
z_zenith = __; %Zenith(rad)
Gamma_azimuth = __; %Azimuth(rad)
P = 0.995609365856; %millibars : Annual average local pressure
T = 12.3333; %deg._Celcius : Annual average local temperature
alpha = __; %Geocentric right ascension
delta = __; %Declination

% Nonlinear equations-----
F(1) = (---equation#1---);
F(2) = (---equation#2---);
end

%Calling function Sol_trust_region
fun = @Sol_trust_region;
x0 = D2R*[Latitude_input, Longitude_input];
options = optimoptions('fsolve', 'Display', 'off');
[x, fval, exitflag, output] = fsolve(fun, x0, options)
```

```
%Output  
x = Latitude_output,Longitude_output
```

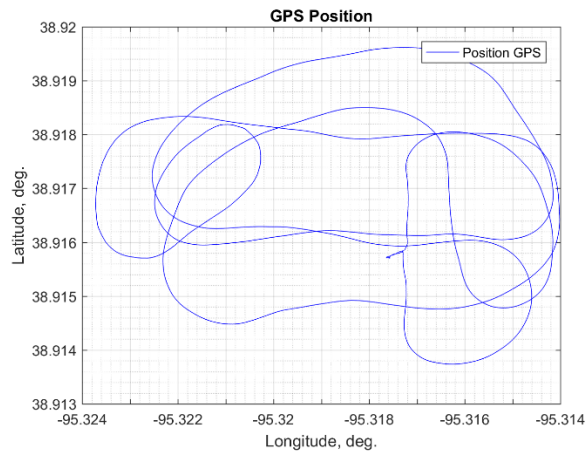
The trust region method has performed very well in solving the derived equations and the result analysis has been performed on ten data sets. For the discussion, only one data set results have been shown and remaining results can be found in **Appendix A**, **Appendix B** and **Appendix C**. The GPS flight data from DG808 unmanned aircraft as shown in Figure 5.3 has been used to mimic the sensor readings for azimuth and zenith angles (sun's position). The GPS time, longitude, latitude, annual average local pressure, and temperature have been used to achieve the azimuth and zenith data from ENEA algorithm. Then this azimuth and zenith angle has been fed as an input towards the implemented algorithm where only the first value (initial value) of azimuth and zenith has been used and the time is used as continuous input. The algorithm needs only the initial value of azimuth and zenith angles, and it uses time dependent remaining parameters of the ENEA algorithm for estimation of next value for azimuth and zenith. The results of the algorithm are discussed as follows.



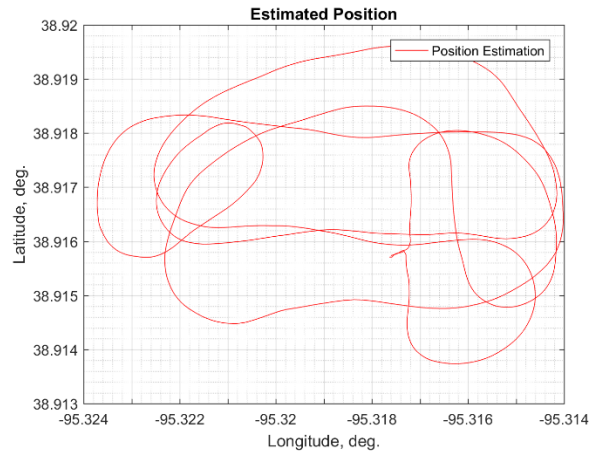
Figure 5.3. DG808 unmanned aircraft

5.2.1 Output of the Trust Region Algorithm

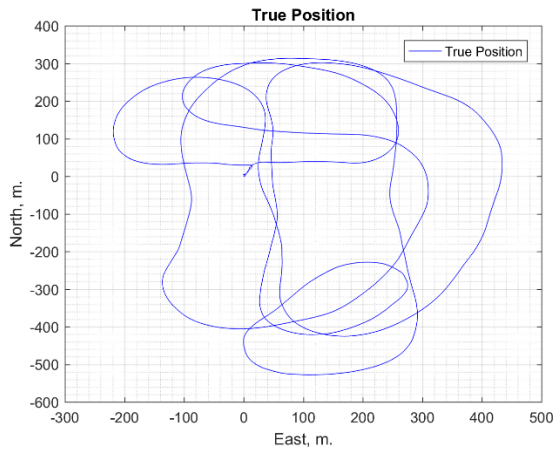
In navigation algorithms where the main objective is to control the position in order to do perform movements in the correct direction, the accuracy of an algorithm can be tested only on basis of position accuracy. The proposed algorithm has provided excellent results with position accuracy. From Figure 5.4, Figure 5.5 and Figure 5.6, it can be observed that the algorithm has followed the flight path with good estimation accuracy.



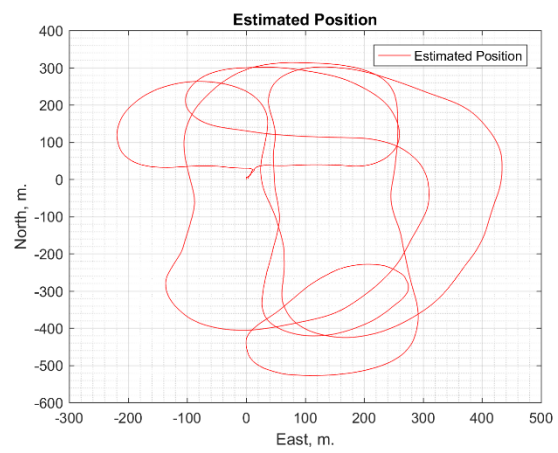
(a)



(b)

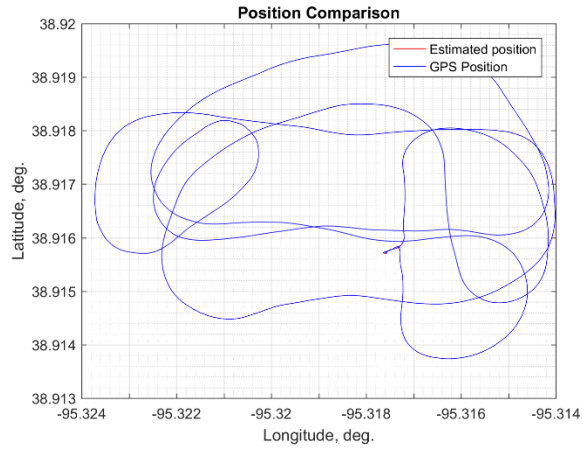


(c)

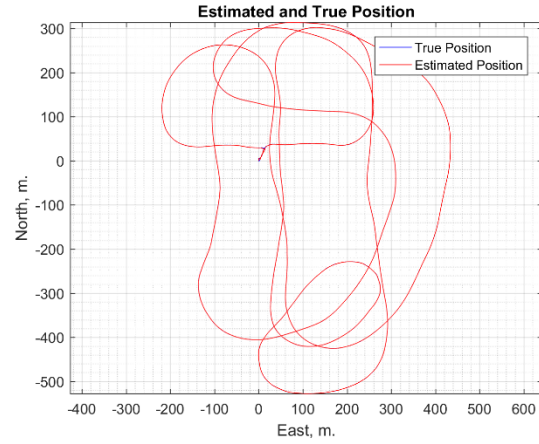


(d)

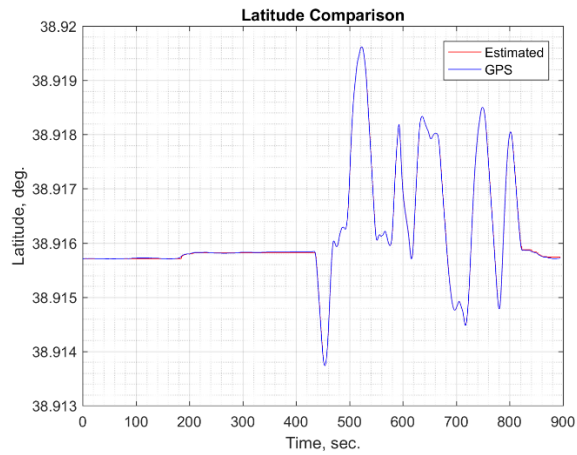
Figure 5.4. True and estimated position (a) GPS position in Geodetic coordinate system, (b) Estimated position in geodetic coordinate system, (c) True position in NED (Local) coordinate system and (d) Estimated position in NED (Local) coordinate system



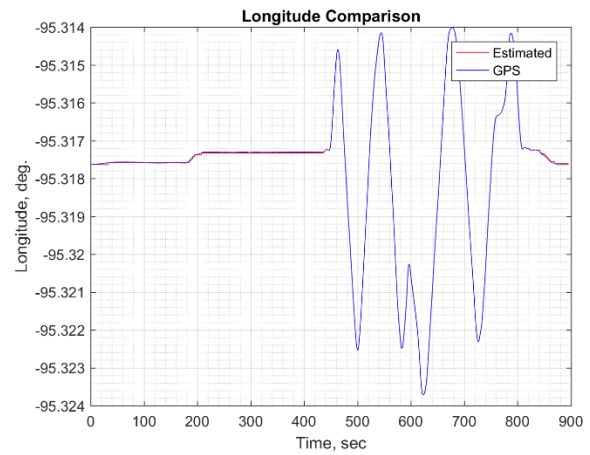
(a)



(b)

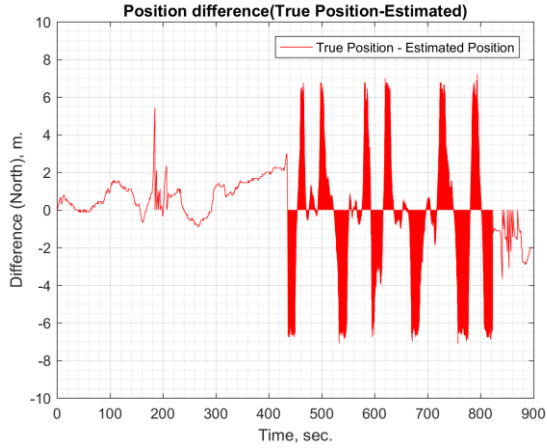


(c)

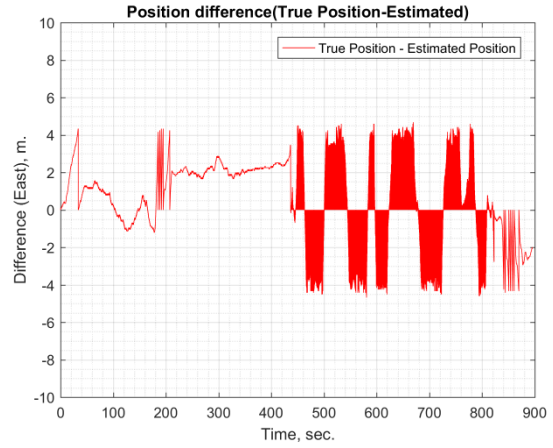


(d)

Figure 5.5. True and estimated position comparison (a) GPS position and estimated position comparison in Geodetic coordinate system, (b) True position and estimated position comparison in NED (Local) coordinate system, (c)) GPS position and estimated position latitude comparison and (d) GPS position and estimated position longitude comparison



(a)



(b)

Figure 5.6. Position difference with respect to time (a) Position difference in North, (b) Position difference in East

The standard deviation in error for the estimation output has been calculated in both North and East direction with reference to local coordinate system. For the data set under discussion, the standard deviation of error in North and East direction is 1.90 meters and 1.92 meters respectively. These numbers are comparable to the performance of GPS units available in the market. The following Table 5.1 also shows the average of the standard deviation of error in North and East direction for the ten data sets used in this study which is 1.5723 meters and 1.4029 meters respectively.

| | Error North Standard Deviation, m. | Error East Standard Deviation m. |
|---------------------|---|---|
| DATA SET #1 | 1.9025 | 1.9167 |
| DATA SET #2 | 0.9119 | 1.1204 |
| DATA SET #3 | 1.0045 | 1.2542 |
| DATA SET #4 | 2.7245 | 2.0589 |
| DATA SET #5 | 2.6415 | 2.8131 |
| DATA SET #6 | 1.9479 | 1.8449 |
| DATA SET #7 | 1.0397 | 1.1777 |
| DATA SET #8 | 1.0724 | 0.6346 |
| DATA SET #9 | 1.4875 | 0.2202 |
| DATA SET #10 | 0.9902 | 0.9885 |
| Average | 1.5723 | 1.4030 |

Table 5.1. Comparison and average of error in North and East standard deviation

5.2.2 Correlation of the Error

The proposed algorithm uses the position of the sun as an input, therefore, the accuracy of algorithm estimation output depends on the availability of sun during a particular time of the day so it becomes necessary to find the correlation of the error. It is essential to establish the relationship with the error based on time of day and day. The relations found are as follows.

5.2.2.1 Error Caused By Time of Day

The relationship of error caused by the availability of sun is very evident based on the sunrise and sunset. There are various parameters that affect the visibility of sun during the day like weather, eclipse, etc. and the assumption in this analysis is that no such parameters are causing visibility issues during the experiments. The experiments have been performed while changing only the hour during the day in input time. As the proposed algorithm is heavily dependent on the accuracy of time, this makes it essential to analyze.

The results have shown that closer the time is to sunset and sunrise, the error elevates. The experiments have also shown that the maximum accuracy has been achieved by the algorithm is between 16 and 20 hours of UTC. The local time difference of the flight tests location with that of UTC was -6 hours so the local time will be 10 AM to 2 PM. This is a time of day when the sun is usually at its brightest intensity. This phenomenon has been observed across all the data sets and Figure 5.7 is a good representation of this. Appendix B consists more data set results of covariance of error with respect to time of the day.

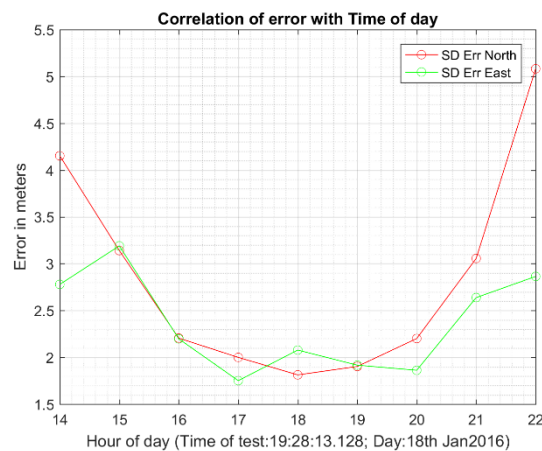


Figure 5.7. Correlation of error with time of day

5.2.2.2 Error Caused By Day

The second parameter to validate is how the error is related to change in the day when the position is fixed. Ideally, it can be easily assumed that if the position is fixed, time is fixed and the only day is changing then the output should be same but this has been proven wrong. There is a variation present in the error for both North and East directions even though the standard deviation of the change in error is less than 0.5 meters in all cases. This can be attributed to the phenomenon called analemma. If you observe the sun from the same place and same time every day, the sun won't be at the same position. Figure 5.8 shows the variation based on the day.

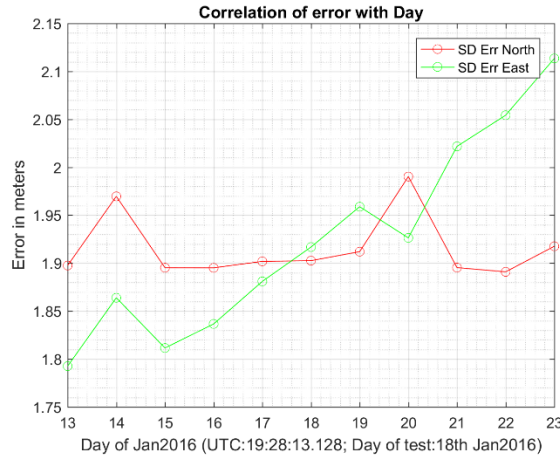


Figure 5.8. Correlation of error with day

5.2.2.3 Error Correlation with Aircraft Heading and Attitude

The correlation of error with aircraft heading and attitude needs to be established and to find the relation, the position difference (true position – estimated position) with respect to time has been calculated. It is important to note that the flight data used for this analysis is only the GPS position and time data to mimic the sensors and no actual sensors were used. The real-life sensors will have some inaccuracies which may limit the roll or other maneuvers of the aircraft to achieve the results for this algorithm.

The true position, estimated position and their comparison of selected data set have been shown in Figure 5.9. To establish the relationship of error with aircraft heading and attitude, first, the relationship of error with the altitude and true velocity needs to be verified. Figure 5.10 contains the comparison of error/position difference (True position -Estimated position) with the altitude (m) and true velocity (m/s) of aircraft. From the plot, it can be concluded that the error in the position starts at the beginning of takeoff, and ends with the landing run. Now the focus of analysis can be shifted towards the duration of flight when the aircraft is airborne.

The relationship of position error with changes in the longitude and latitude can be established even though they both are in different coordinate systems as time is the common parameter. The comparison has been shown in Figure 5.11, which contains the comparison of GPS and estimated latitude and longitude along with errors in north and east (local coordinate system). The comparison shows that error in north follows the changes in latitude and error in east follows the error in longitude.

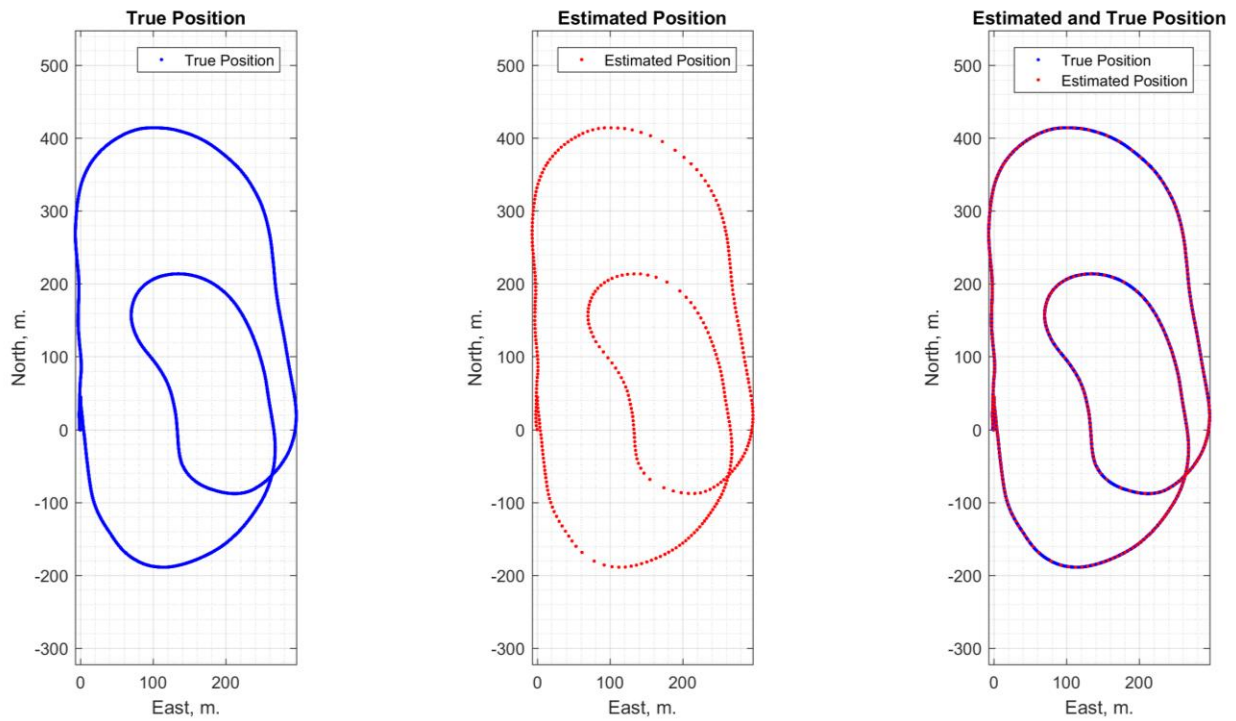


Figure 5.9. True position, estimated position and comparison

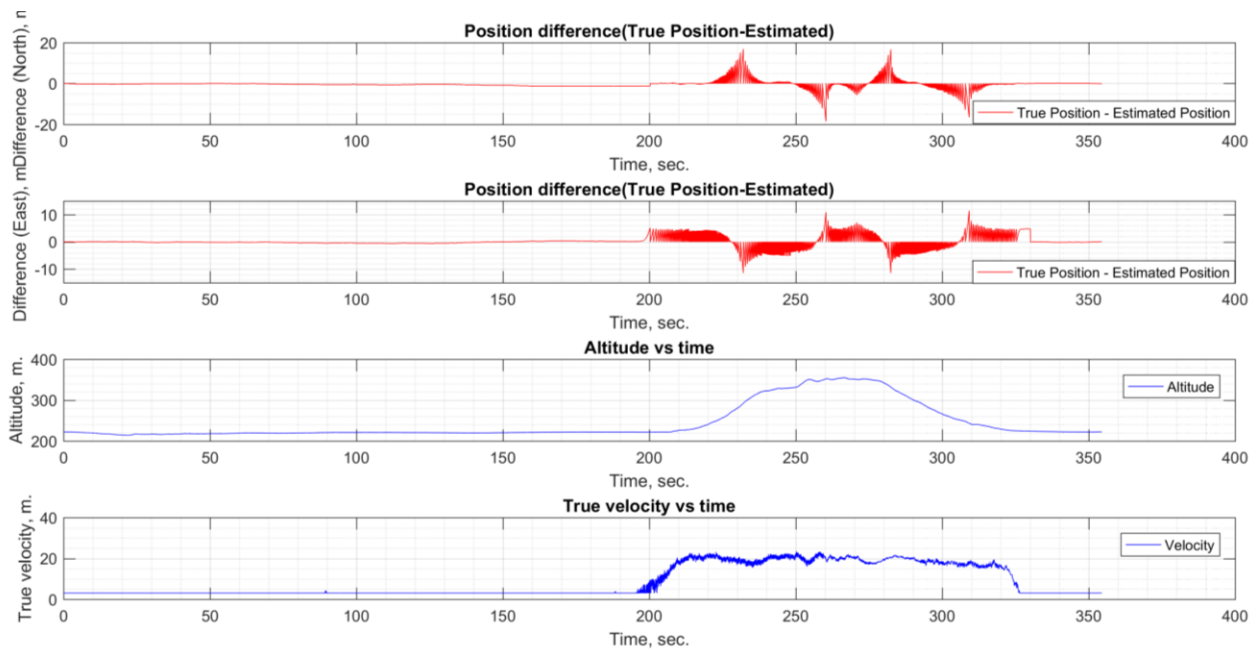


Figure 5.10. Position difference, Altitude and True velocity

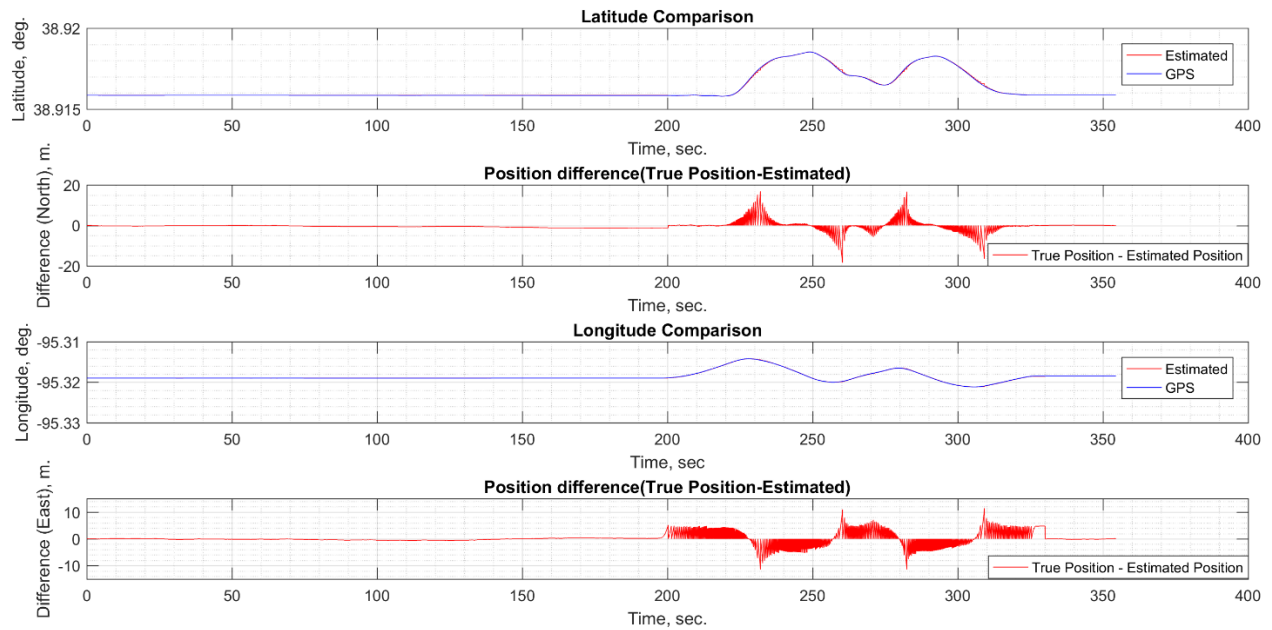
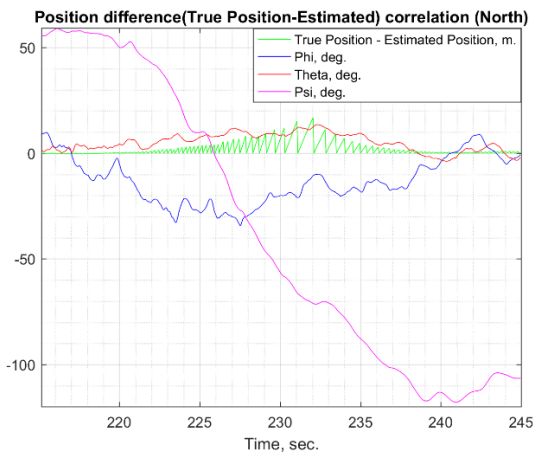
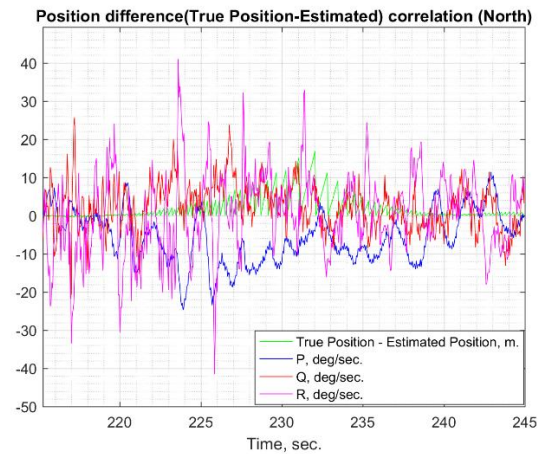


Figure 5.11. Position difference, Latitude comparison and Longitude comparison



(a)



(b)

Figure 5.12. Correlation of error (a) Correlation of error with roll, pitch and yaw, (b) correlation of error with roll rate, pitch rate and yaw rate

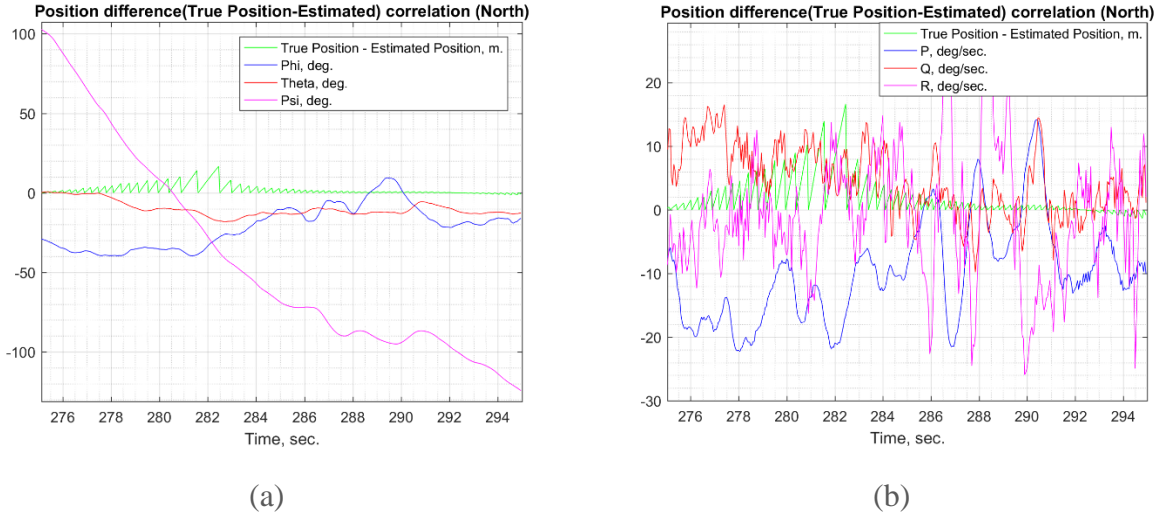
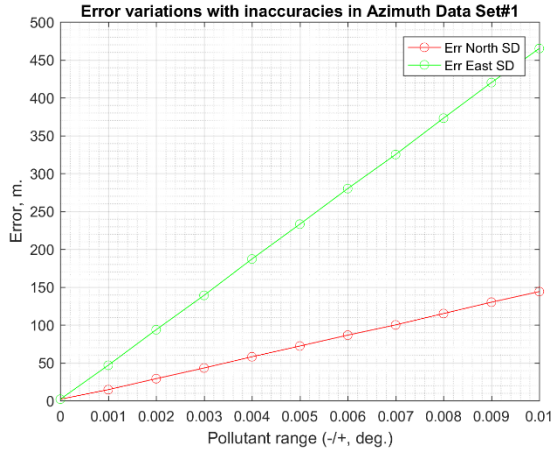


Figure 5.13. Correlation of error (a) Correlation of error with roll, pitch and yaw, (b) correlation of error with roll rate, pitch rate and yaw rate

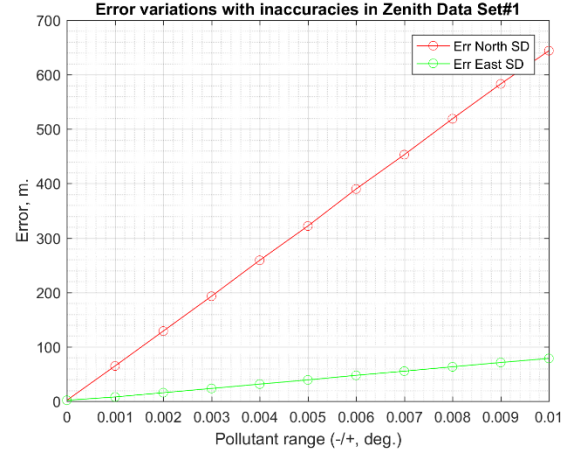
The correlation of error with the attitude of aircraft can be established with the comparison of error with aircraft attitude parameters (ϕ, θ, ψ) and 3D rate gyro parameters (P, Q, R). Each attitude parameter will have some impact on the output, the comparison can only demonstrate which parameter has a more visible impact on the output. Figure 5.12 (a) and Figure 5.13 (a) contains the comparison of attitude angles (roll, pitch, yaw) with the error and Figure 5.12 (b) and Figure 5.13 (b) contains the comparison of rate gyro parameter (roll rate, pitch rate, yaw rate) with the error. It can be observed from Figure 5.12 and 5.13, that error increases with increase in negative roll and roll rate also when the roll and roll rate goes from positive to negative, the error is small.

5.2.3 Investigation of Sensitivity of Error Caused by Sensors

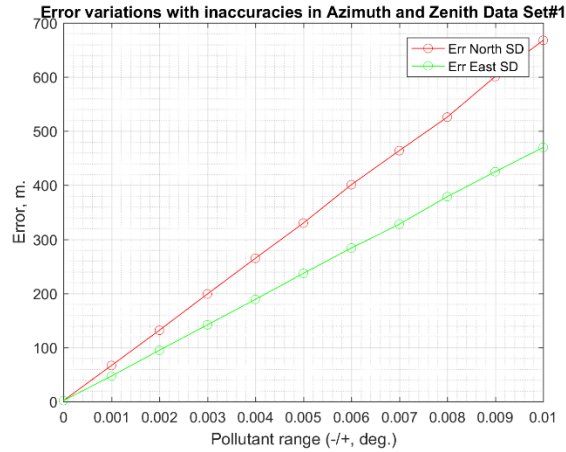
The real-life sensors always have some inaccuracies thus it becomes essential to study the sensitivity of proposed algorithm towards the errors caused by the sensor to validate the performance of the algorithm. The sensitivity investigation has been performed by polluting the input data of azimuth and zenith angles to simulate the sensors. Figure 5.14 is the representation of the results where the error is measured with respect to the pollutant range. This study has been conducted by first polluting the azimuth data, secondly the zenith data and finally both azimuth and zenith data as shown in Figure 5.14 (a), (b) and (c) respectively.



(a)



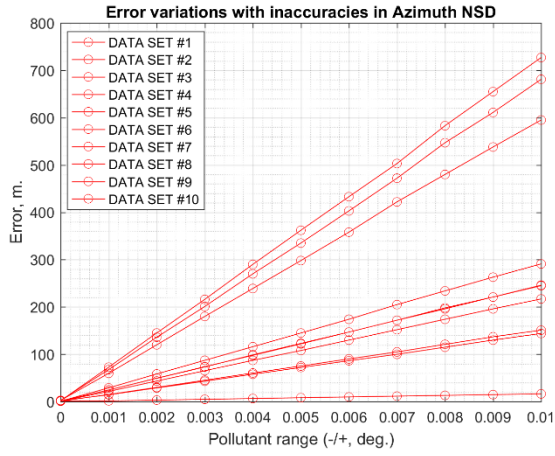
(b)



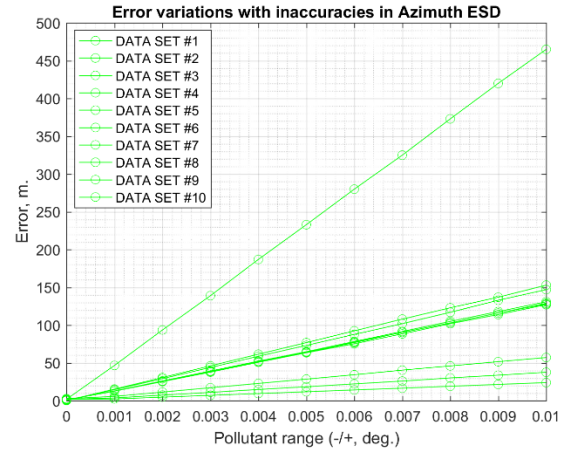
(c)

Figure 5.14. Error variations with respect to inaccuracies in sensor data (a) Error variation with inaccuracies in Azimuth, (b) Error variation with inaccuracies in Zenith and (c)) Error variation with inaccuracies in Azimuth and Zenith

In all the experiments, at 0.01° pollutant angle, the error standard deviation in North or East has met the boundaries of the area of operation of that flight path or exceeded it. The current example, the area of operation is approximately 800×600 meters. From Figure 5.14, the maximum error is 475 meters in East for azimuth pollutant, 650 meters in North for zenith pollutant and 650 meters for both azimuth and zenith pollutant. As the standard deviation of error crosses the area of operation boundary or gets close to it, the algorithm becomes ineffective for the estimation. This shows the algorithm is highly sensitive to the errors in the input. The experiment results for all the data sets have been combined in Figure 5.15, Figure 5.16 and Figure 5.17 to give a clear idea about the sensitivity of the algorithm.

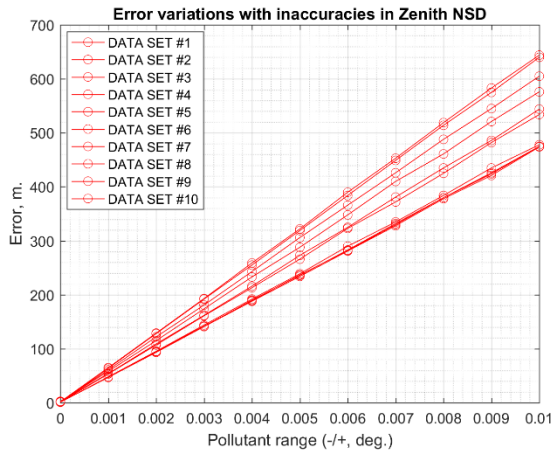


(a)

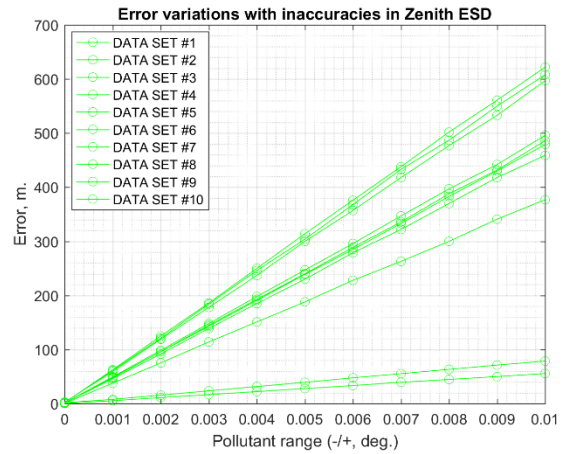


(b)

Figure 5.15. Error variations with respect to inaccuracies in Azimuth for 10 data sets (a) Error variations in North standard deviation with respect to inaccuracies in Azimuth, (b) Error variations in East standard deviation with respect to inaccuracies in Azimuth

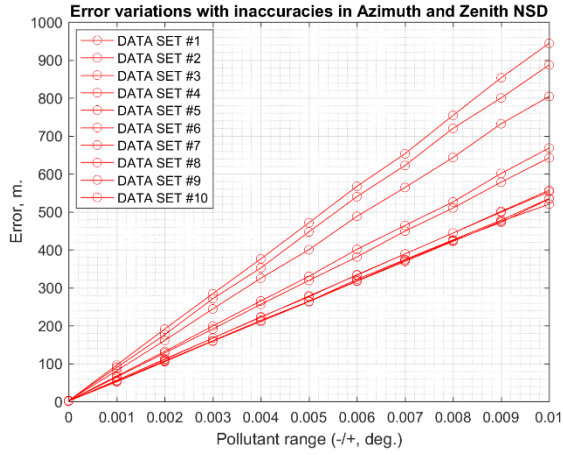


(a)

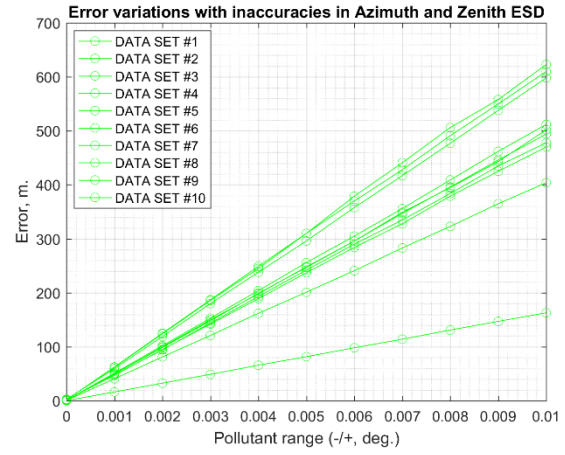


(b)

Figure 5.16. Error variations with respect to inaccuracies in Zenith for 10 data sets (a) Error variations in North standard deviation with respect to inaccuracies in Zenith, (b) Error variations in East standard deviation with respect to inaccuracies in Zenith

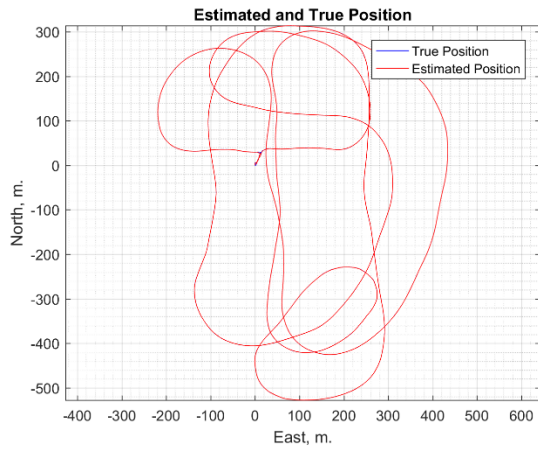


(a)

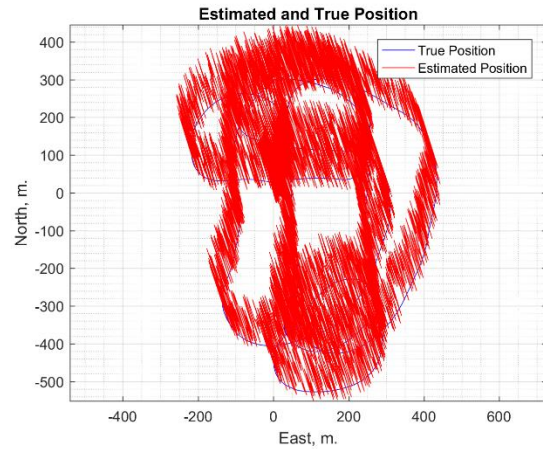


(b)

Figure 5.17. Error variations with respect to inaccuracies in Azimuth and Zenith for 10 data sets (a) Error variations in North standard deviation with respect to inaccuracies in Azimuth and Zenith, (b) Error variations in East standard deviation with respect to inaccuracies in Azimuth and Zenith



(a)



(b)

Figure 5.18. Estimated and true position (a) Estimated and true position with no pollutant, (b) Estimated and true position with 0.001° pollutant in azimuth angle.

The sensitivity of the algorithm can also be demonstrated as shown in Figure 5.18, the plots show the difference between the estimated and true position of data set with no pollutant and with the pollutant of 0.001° in azimuth angle input. The sensitivity makes this algorithm ineffective for use with the currently available sensors due to the level of errors present in the calculation of the

sun's position. The standard deviation of error present is 14.4253 meters towards the north and 46.6077 meters towards east.

One of the most accurate sensors for calculation of zenith has an accuracy of 0.3° and to verify the amount of error that can be reached with this accuracy the pollutant of 0.3° has been added to the zenith input of the algorithm.

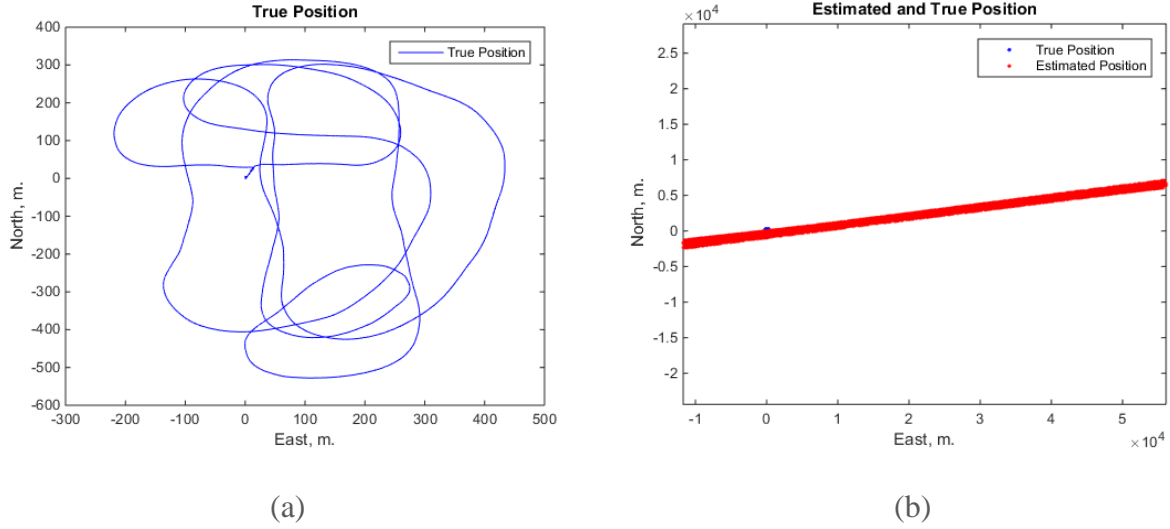


Figure 5.19. Estimated and true position (a) True position with no pollutant, (b) Estimated and true position with 0.3° pollutant in azimuth angle.

The Figure 5.19 shows the comparison of true position with the estimated position by the algorithm for the error input of 0.3° . The standard deviation of error in north direction is 19406.9366 meters and in east direction is 2368.5836 meters i.e. 19.4070 Km and 2.3686 Km respectively. This effectively proves that the available sensors cannot be used as inputs for the proposed algorithm due to the insufficient accuracies.

5.2.4 Miscellaneous

5.2.4.1 Behavior of algorithm

The behavior of the algorithm can be studied by studying the error pattern. The observation shows that error oscillated between zero and some value throughout the study and this phenomenon has everything to do with the solution received from the algorithm. As shown in following Figure 5.20, at first glance it looks like the density of points in output has reduced but the convergence or output of position between these points has saturated (same solution for multiple points) to a point and it jumps to a next when it finds a sufficient difference in input.

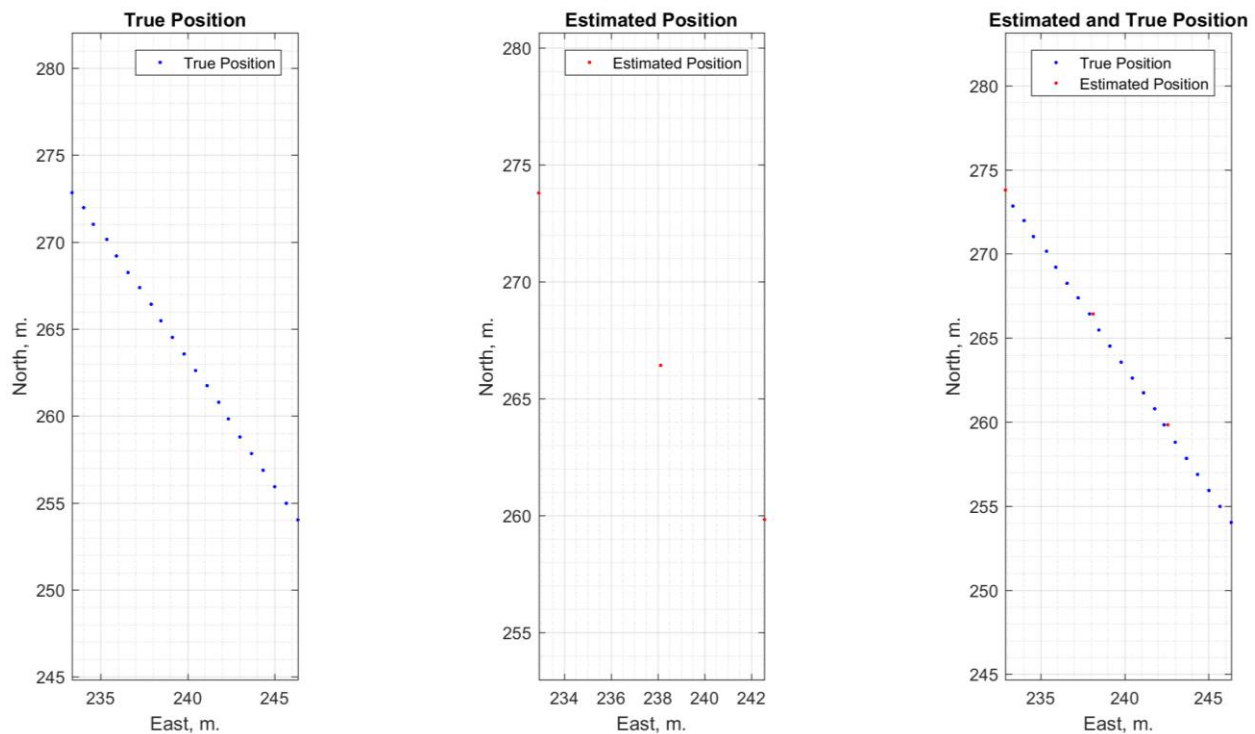


Figure 5.20. True position, estimated position and comparison

The exact value the algorithm needs to react for next point could not be found as it varies throughout. The error is more in some part of the path when there is a turn (in terms of 2D position and not in terms of the attitude of aircraft), as visible in following Figure 5.21. It was found that that error increases with increase in negative roll and roll rate as presented in section 5.2.2.3.

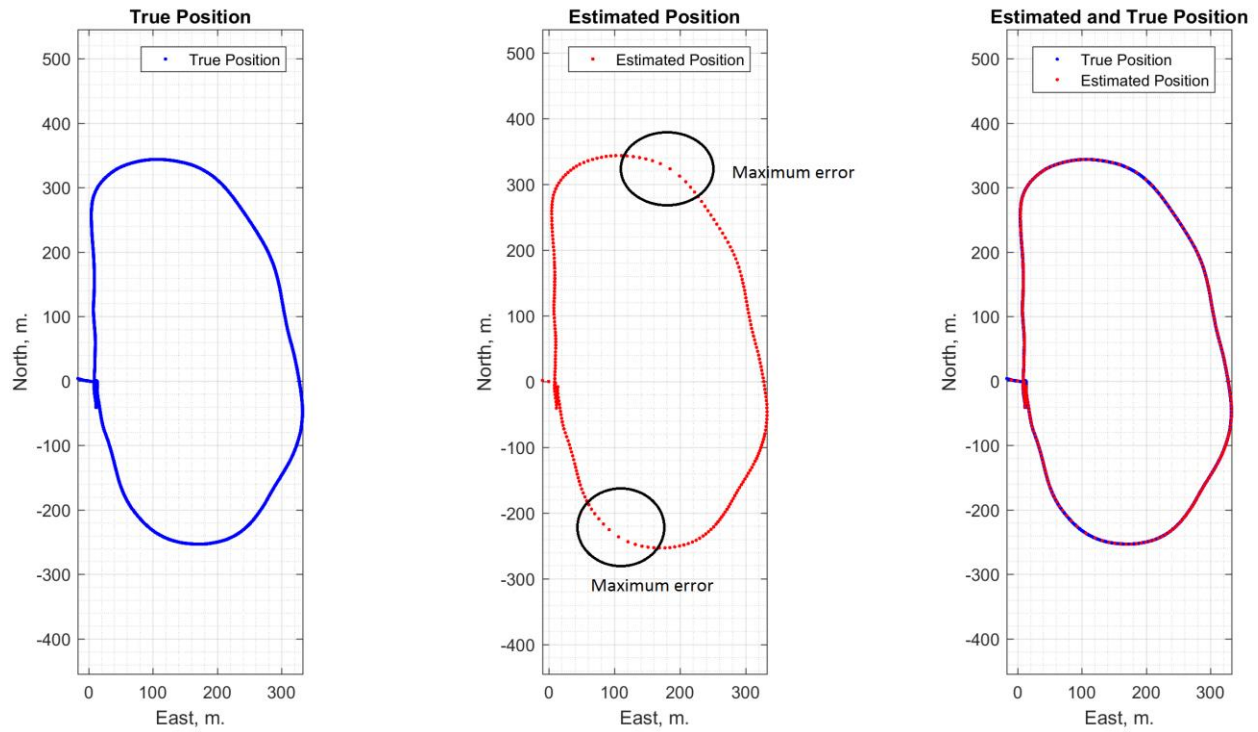


Figure 5.21. True position, estimated position and comparison

5.2.4.2 Execution time and data rate of algorithm

The feasibility of algorithm to be used in the real-life application depends on the data rate. The calculation of execution time and data rate based on the data points have been conducted and the minimum data rate achieved in 4.8998 Hz.

| Data Set # | Data points | Execution time, Sec. | Data rate, Hz |
|------------|-------------|----------------------|---------------|
| 1 | 17904 | 2995.49 | 5.9769 |
| 2 | 11165 | 992.63 | 11.2478 |
| 3 | 9673 | 865.30 | 11.1787 |
| 4 | 5181 | 243.52 | 21.2754 |
| 5 | 9046 | 787.80 | 11.4826 |
| 6 | 7090 | 412.60 | 17.1837 |
| 7 | 14389 | 2936.61 | 4.8998 |
| 8 | 10494 | 1080.08 | 9.7159 |
| 9 | 13635 | 1776.90 | 7.6734 |
| 10 | 11871 | 1394.38 | 8.5134 |

Table 5.2. Execution time and data rate of algorithm

The data used in this project is from the actual flight test and the data rate of the GPS receiver used on board is 4 Hz. This indicates that the presented algorithm is feasible for use in real-life applications. Table 5.2 shows the execution time and data rate for all the ten data sets used in this study.

5.2.4.3 Accuracy of GPS at stationary position

One of the major assumptions for the study performed was that the GPS position data received from the unmanned aircraft was accurate. However, the GPS receivers have their own set of errors and they are not accurate. The major error terms contributing to the positioning error in GPS satellite navigation are atmospheric effects (Ionosphere and troposphere), ephemeris and clock errors, selective availability (historical), receiver noise and multipath errors. The correctable errors are ephemeris and clock errors and atmospheric effect errors (Ionosphere and troposphere). [21]

For the accuracy analysis of GPS systems, one study was conducted before this project. In the study, two GPS receivers (u-blox LEA6H and NovAtel OEM615) were used for a comparative analysis at a stationary location. Two reference points were selected in north direction with known distance between them. It was also made sure in the selection process that these points were clearly visible on the map so that the points are accessible and position data in terms of longitude, latitude and altitude can be collected from selected location. The selected location and reference points are shown in Figure 5.22. The longitude and latitude readings provided the distance between those selected points as 5.35 meters; however, the actual distance between these two points was found to be 5.15 meters when it was physically calculated.



Figure 5.22. The location and setup

Ideally, the position data received should not show any changes in position as the GPS receiver is stationary. It was observed that the GPS position readings bounces around rather than showing its actual stationary position. That means, all the readings apart from original location are the errors in the position so the mean and standard deviation of the errors were calculated as shown in Table 5.3. The distance of mean of locations from the original location was calculated and found to be varying between 1.5882 meters to 2.2939 meters for u-blox GPS receiver and 0.4693 meters to 1.3071 meters for NovAtel GPS receiver. In the comparative study, the accuracy of NovAtel receiver was better than that of the u-blox receiver.

| Parameter | u-blox 1 | u-blox 2 | u-blox 3 | NovAtel 1 | NovAtel 2 | NovAtel 3 |
|---|-----------------|-----------------|-----------------|------------------|------------------|------------------|
| N position mean, (m) | 2.3875 | 1.7953 | 1.5107 | 0.1074 | 0.8456 | 1.0046 |
| N position S.D. , (m) | 0.6445 | 0.9351 | 0.6637 | 0.9316 | 1.0109 | 0.6204 |
| E position mean, (m) | -0.1712 | -0.2699 | -0.4900 | 0.1521 | 0.1502 | 0.5315 |
| E position S.D. , (m) | 0.5996 | 0.4751 | 0.4564 | 0.2549 | 0.3627 | 0.3376 |
| D position mean, (m) | -1.1147 | 3.6279 | 2.7034 | -1.1195 | 0.9536 | 0.6054 |
| D position S.D. , (m) | 1.4356 | 1.1605 | 1.2112 | 1.9078 | 0.9544 | 1.1635 |
| hMSL position mean, (m) | 283.11 | 278.37 | 279.30 | 283.12 | 281.05 | 281.39 |
| hMSL position S.D. , (m) | 1.4356 | 1.1605 | 1.2112 | 1.9079 | 0.9544 | 1.1634 |
| Distance from true position, (m) | 2.2939 | 1.8155 | 1.5882 | 0.4693 | 0.9602 | 1.3071 |

Table 5.3. Performance comparison of u-blox LEA6H and NovAtel OEM615

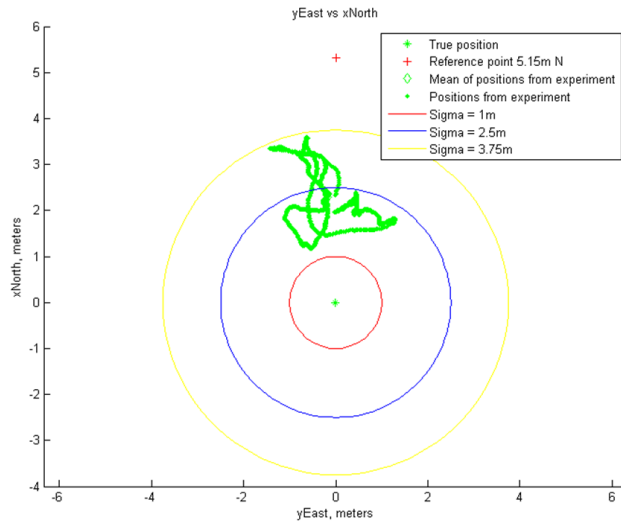


Figure 5.23. u-blox on NE frame accuracy

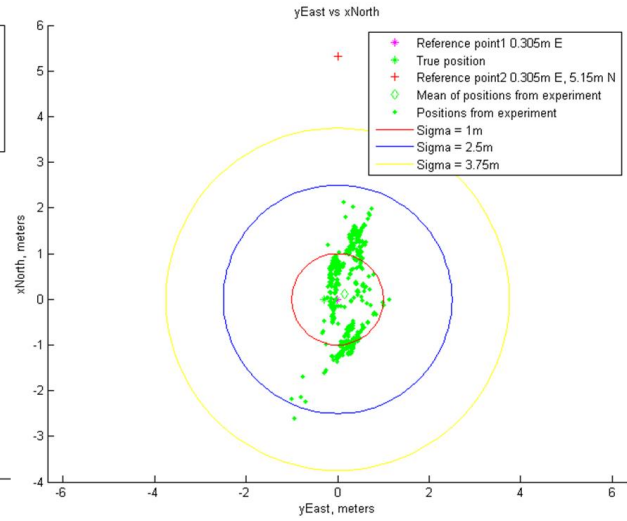


Figure 5.24. NovAtel on NE frame accuracy

For this study, decoder programs were developed to decode the data from GPS receivers and the position data was plotted to visualize the accuracy. Figures 5.23 and 5.24 shows the position readings for one of the data set along with true position, mean of positions from readings and reference point located at 5.15 meters at north.

Chapter 6

Future work

The sensitivity analysis results of this algorithm create immense scope for improvements. One of the major concern during this study has been whether the algorithm will be able to recover when the sensors are blindsided. The algorithm will be able to recover from blinding considering the last reasonable value is used as initial condition again as the remaining parameters are based on time. However, the time/distance escaped in recovery should be very small otherwise, the trust region algorithm will not be able to converge the values towards accurate estimation. The maximum time/distance value from where the presented algorithm will be able to recover has varied over the study and showed dependence on the current flying pattern of the aircraft. The blinding effect can also be addressed using multiple sensor suits to cover the complete spectrum which can withstand attitude of the aircraft as discussed in Chapter 3. Sensors. At this point, the algorithm cannot withstand the large errors caused by changes in attitude of aircraft as discussed in sensitivity analysis.

The use of extended Kalman filter can bring the necessary robustness in the algorithm. The recovery from blinding effect or any spikes in input sensor values can be better handled by this approach. Using, extended Kalman filter (EKF [10]) the state vector and Jacobians can be defined as follows and it is important to note that the initial value of x_0 will be the last known value of observer's longitude (θ) and latitude (φ).

Time update:

The state vector will be in terms of θ and φ and can be given as follows,

$$\hat{x}_{k|k-1} = \begin{bmatrix} f(\theta_{k-1|k-1}) \\ f(\varphi_{k-1|k-1}) \end{bmatrix} \quad (\text{XII})$$

($\hat{x}_{k|k-1}$ Contains dynamic model of θ and φ in terms of Z and γ)

The Jacobian of the state vector can be defined as follows,

$$A_k = \begin{bmatrix} \frac{\partial f(\theta_{k-1|k-1})}{\partial \theta} & \frac{\partial f(\theta_{k-1|k-1})}{\partial \varphi} \\ \frac{\partial f(\varphi_{k-1|k-1})}{\partial \theta} & \frac{\partial f(\varphi_{k-1|k-1})}{\partial \varphi} \end{bmatrix} \quad (\text{XIII})$$

Measurement update:

The measurement state, Jacobian and the correction state can be defined as follows,

$$y_k = h(\hat{x}_{k|k-1}) = \begin{bmatrix} f(Z_{k|k-1}) \\ f(\gamma_{k|k-1}) \end{bmatrix} \quad (\text{XIV})$$

(y_k - Calculated)

$$H_k = \begin{bmatrix} \frac{\partial f(Z_{k|k-1})}{\partial \theta} & \frac{\partial f(Z_{k|k-1})}{\partial \varphi} \\ \frac{\partial f(\gamma_{k|k-1})}{\partial \theta} & \frac{\partial f(\gamma_{k|k-1})}{\partial \varphi} \end{bmatrix} \quad (\text{XV})$$

$$z_k = \begin{bmatrix} (Z_k) \\ (\gamma_k) \end{bmatrix} \quad (\text{XVI})$$

(z_k - Direct readings)

Using equations from (VIII) to (XII) in the EKF algorithm as described above we can get the solution of nonlinear systems we have in real time.

The whole idea behind this project is to present an alternative to the GPS positioning system. A major chunk of aircraft these days still uses traditional dead reckoning method for navigation which uses the 3D accelerometer (a_x, a_y, a_z), 3D rate gyro (P, Q, R) and a pressure sensor (V_a). The GPS positioning system has been an addition to this traditional method so if we want to replace GPS positioning system we will still need to use the above-mentioned sensors and correct the errors in positioning by integrating the position found from the solar position. The following equation will be the second state vector which contains inertial/local position (P_n, P_e, P_d), body frame velocities (U, V, W) and aircraft attitude parameters (ϕ, θ, φ).

$$\hat{x}_{k|k-1} = \begin{bmatrix} f(P_{n_{k-1}|k-1}) \\ f(P_{e_{k-1}|k-1}) \\ f(P_{d_{k-1}|k-1}) \\ f(U_{k-1|k-1}) \\ f(V_{k-1|k-1}) \\ f(W_{k-1|k-1}) \\ f(\phi_{k-1|k-1}) \\ f(\theta_{k-1|k-1}) \\ f(\varphi_{k-1|k-1}) \end{bmatrix} \quad (\text{XVII})$$

The state vector can further be elaborated as follows [20] [8],

$$\begin{bmatrix} \dot{P}_n \\ \dot{P}_e \\ \dot{P}_d \\ \dot{U} \\ \dot{V} \\ \dot{W} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} (\cos\theta * \cos\varphi)U + (\sin\phi * \sin\theta * \cos\varphi - \cos\phi * \sin\varphi)V + (\cos\phi * \sin\theta * \cos\varphi + \sin\phi * \sin\varphi)W \\ (\cos\theta * \sin\varphi)U + (\sin\phi * \sin\theta * \sin\varphi + \cos\phi * \cos\varphi)V + (\cos\phi * \sin\theta * \sin\varphi - \sin\phi * \cos\varphi)W \\ (\sin\theta)U - (\sin\phi * \cos\theta)V - (\cos\phi * \cos\theta)W \\ R * V - Q * W - g * \sin\theta + a_x \\ P * W - R * U + g * \cos\theta * \sin\phi + a_y \\ Q * U - P * V + g * \cos\theta * \cos\phi + a_z \\ P + (\sin\phi * \tan\theta)Q + (\cos\phi * \tan\theta)R \\ (\cos\phi)Q - (\sin\phi)R \\ (\sin\phi * \sec\theta)Q + (\cos\phi * \sec\theta)R \end{bmatrix} \quad (\text{XVIII})$$

The idea behind two parallel state vectors/executions in the same extended Kalman filter is, two state vectors are independent of each other's influence but they execute at the same iterations. However, these two parallel flows need to communicate for error correction. The Y and Z of first flow (equation X and XII respectively) will need to be converted from zenith and azimuth to the longitude and latitude to keep the state vector in the same coordinate system. This correction in the longitude and latitude frame can then be converted to the local/inertial NED frame to acts as the error correction for the second flow local position coordinates. This way the first flow model will act as only the observer (error correction/sensor) for the second flow which has the basic traditional sensors i.e. 3D accelerometer, 3D rate gyro.

Chapter 7

Conclusion

In this study, a bio-inspired navigation algorithm has been developed with the help of the ENEA algorithm. The proposed algorithm has been implemented using the trust region method. It has been found that if the position of sun in terms of azimuth and zenith; and accurate time is available then the position of an observer can be found with very good accuracy. The average standard deviations of the error in the North and East directions for the ten data sets used in this study, have been found to be 1.57 meters and 1.40 meters, respectively. However, these results are for the ideal case when the assumed sensor data has no errors present. To validate the algorithm further, sensitivity tests have been performed where the input sensor data has been polluted to mimic real sensor data. In all the experiments, with 0.01° pollutant angle the error standard deviations in North or East came close to the boundaries of area of operation of that flight path or exceeded it. As the standard deviation of error crosses the area of operation boundary or gets close to it, the algorithm becomes ineffective for the estimation. This shows the algorithm is highly sensitive to the errors in the input. The most accurate sensors out of the sensors presented in chapter 3; gives accuracy of less than 0.3° and precision of less than 0.05° , which exceeds the maximum error tolerance level of the presented algorithm.

This algorithm uses solar position as input so the correlation of error with respect to the time of day and day have been performed. When the tests were performed to verify the effect of day on accuracy, it was found that there is a variation present in the error for both North and East directions but the standard deviation of the change in error is found to be less than 0.5 meters in all cases. The results for the effect of time of day on accuracy have shown that the error increased closer to the time of sunset and sunrise. The experiments also show the maximum accuracy achieved by algorithm was between 16 and 20 hours of UTC which is 10 AM to 2 PM local time.

From the study, it can be concluded that when very accurate position of sun and time is known, the position of an observer can be calculated. However, with the current available sensors it is not possible to implement this algorithm practically which means at this point the presented algorithm cannot be used to replace GPS systems. If in future the sensor technology advances enough to provide acceptable inputs to the presented algorithm, this approach has immense uses. The same approach can be used for any planet in this solar system provided the solar position calculation algorithm designed for that planet has the required accuracy and the sun is visible.

References

- [1] Roberto Grena, ‘An algorithm for the computation of the solar position’, Solar Energy 82 (2008) 462-470, Elsevier Academic Press.
- [2] Ibrahim Reda, Afshin Andreas, ‘Solar Position Algorithm for Solar Radiation Applications’, NREL/TP-560-34302, National Renewable Energy Laboratory.
- [3] Hans G. Wallraff, ‘Avian Navigation: Pigeon Homing as a Paradigm’, Chapter 5 and Chapter 6, ISBN 3-540-22385-1, Springer Publications.
- [4] Randolph Menzel, Uwe Greggers, ‘The memory structure of navigation in honeybees’, J Comp Physiol A (2015) 201:547-561, Springer Publications.
- [5] Julia R. Ashkanazy, J. Sean Humbert, ‘Bio-Inspired Absolute Heading Sensing Based on Atmospheric Scattering’, AIAA SciTech; AIAA Guidance, Navigation and Control Conference.
- [6] Joseph J. Michalsky, ‘The Astronomical Almanac’s Algorithm for Approximate Solar Position’, Solar Energy Vol. 40, No. 3, pp 227-235, 1988, Pergamon Journals Ltd.
- [7] Jean Meeus, ‘Astronomical Algorithms’, ISBN 0-943396-35-2, Willmann-Bell, Inc.
- [8] Randal W. Beard, Timothy W. McLain, ‘Small Unmanned Aircraft: Theory and Practice’, ISBN 978-0-691-14921-9, Princeton University Press.
- [9] J. Douglas Faires, Richard Burden, ‘Numerical Methods: Third Edition’, ISBN 0-534-40761-7, Thomson: Brooks/Cole.
- [10] Greg Welch, Gary Bishop, ‘An Introduction to the Kalman Filter’, Updated: Monday, July 24, 2006; TR 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599-3175
- [11] E. Shlizerman, J. Philips-Portillo, D. B. Forger, S. M. Reppert, ‘Time-compensated sun compass in the migrating monarch butterfly’, Cell Reports 2016
- [12] nanoSSOC-A60 Sun Sensor for Nano-Satellites Brochure, SOLAR MEMS TECHNOLOGIES
- [13] nanoSSOC-D60 Sun Sensor for Nano-Satellites Brochure, SOLAR MEMS TECHNOLOGIES
- [14] SSoC-D60 2 AXIS ACCURATE SUN SENSOR Brochure, SOLAR MEMS TECHNOLOGIES

[15] SSOC-D60 2 AXIS ACCURATE SUN SENSOR Brochure, SOLAR MEMS TECHNOLOGIES

[16] Dario Floreano, Ramon Pericet-Camara, Stéphane Viollet, Franck Ruffier, Andreas Brückner, Robert Leitel, Wolfgang Buss, Mohsine Menouni, Fabien Expert, Raphaël Juston, Michal Karol Dobrzynski, Geraud L'Eplattenier, Fabian Recktenwald, Hanspeter A. Mallot, and Nicolas Franceschini, Miniature Curved Artificial Compound Eyes, hal-00835031, version 1- 18 Jun 2013, Researchgate
(*Proceedings of the National Academy of Sciences*, vol. 110, no. 23, pp 9267-9272)

[17] Ya-xiang Yuan, A review of trust region algorithms for optimization, State Key Laboratory of Scientific and Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences, P. O. Box 2719, Beijing 100080, China

[18] Equation solving algorithms, MathWorks
(<http://www.mathworks.com/help/optim/ug/equation-solving-algorithms.html#f51887>)

[19] Andrew R. Conn, Nicholas I. M. Gould, Philippe L. Toint, Trust-Region Methods, MPS-SIAM Series on Optimization, ISBN-13: 978-0898714609

[20] Gonzalo A. Garcia, Shahriar Keshmiri, Integrated Kalman filter for a flight control system with redundant measurements, AIAA 2012-2499 (University of Kansas, Lawrence, KS,66045)

[21] Spilker, James J., Bradford W. Parkinson, and Penina Axelrad. Global Positioning System, edited by James J. Spilker, et al., American Institute of Aeronautics and Astronautics, 1995.

Appendices

Appendix A. Output of Trust region algorithm

DATA SET #1

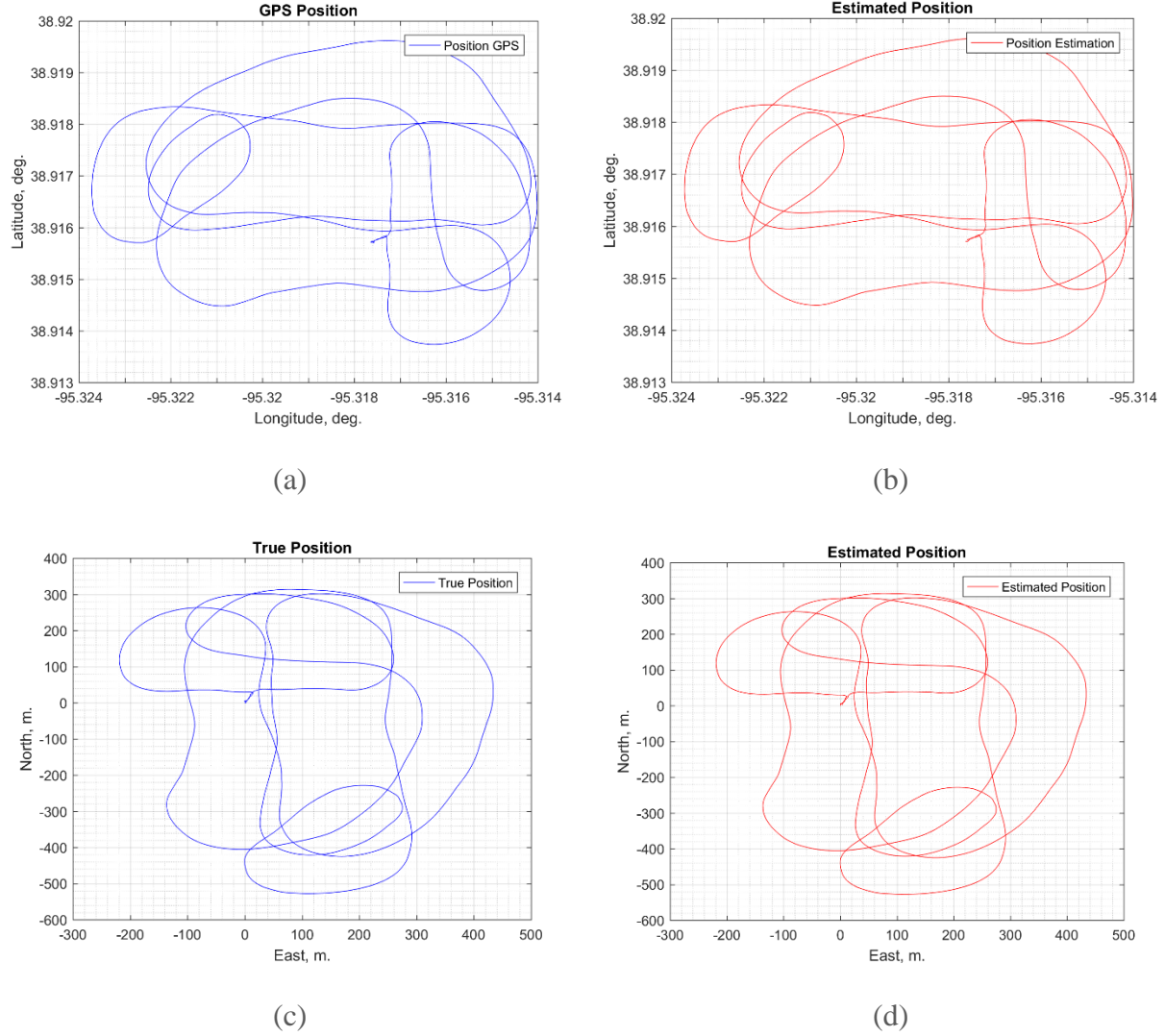
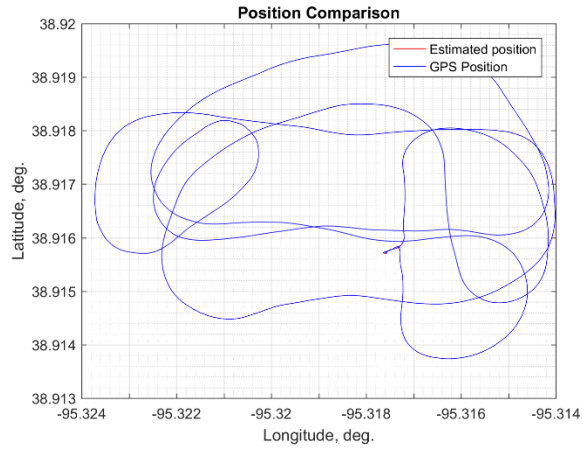
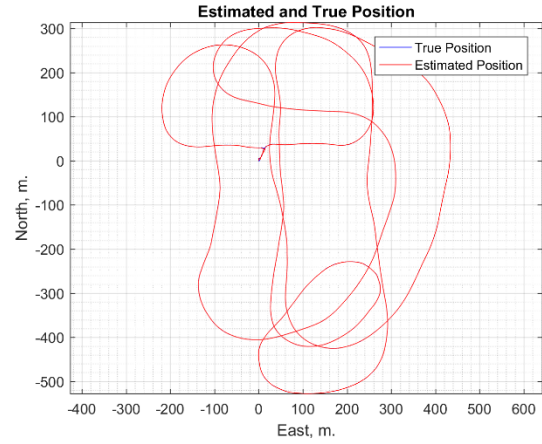


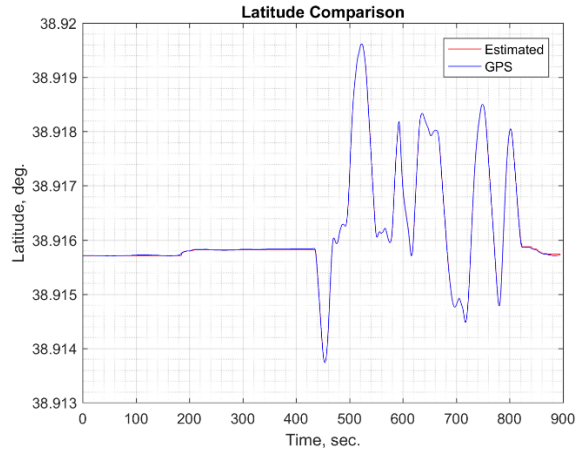
Figure A.1. True and estimated position (a) GPS position in Geodetic coordinate system, (b) Estimated position in geodetic coordinate system, (c) True position in NED (Local) coordinate system and (d) Estimated position in NED (Local) coordinate system



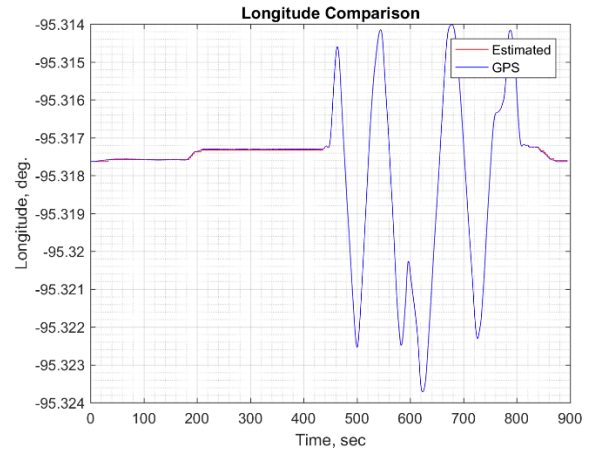
(a)



(b)



(c)

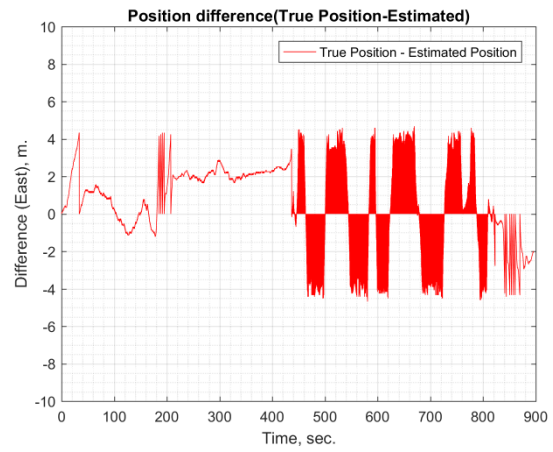


(d)

Figure A.2. True and estimated position comparison (a) GPS position and estimated position comparison in Geodetic coordinate system, (b) True position and estimated position comparison in NED (Local) coordinate system, (c)) GPS position and estimated position latitude comparison and (d) GPS position and estimated position longitude comparison



(a)



(b)

Figure A.3. Position difference with respect to time (a) Position difference in North, (b) Position difference in East

DATA SET #2

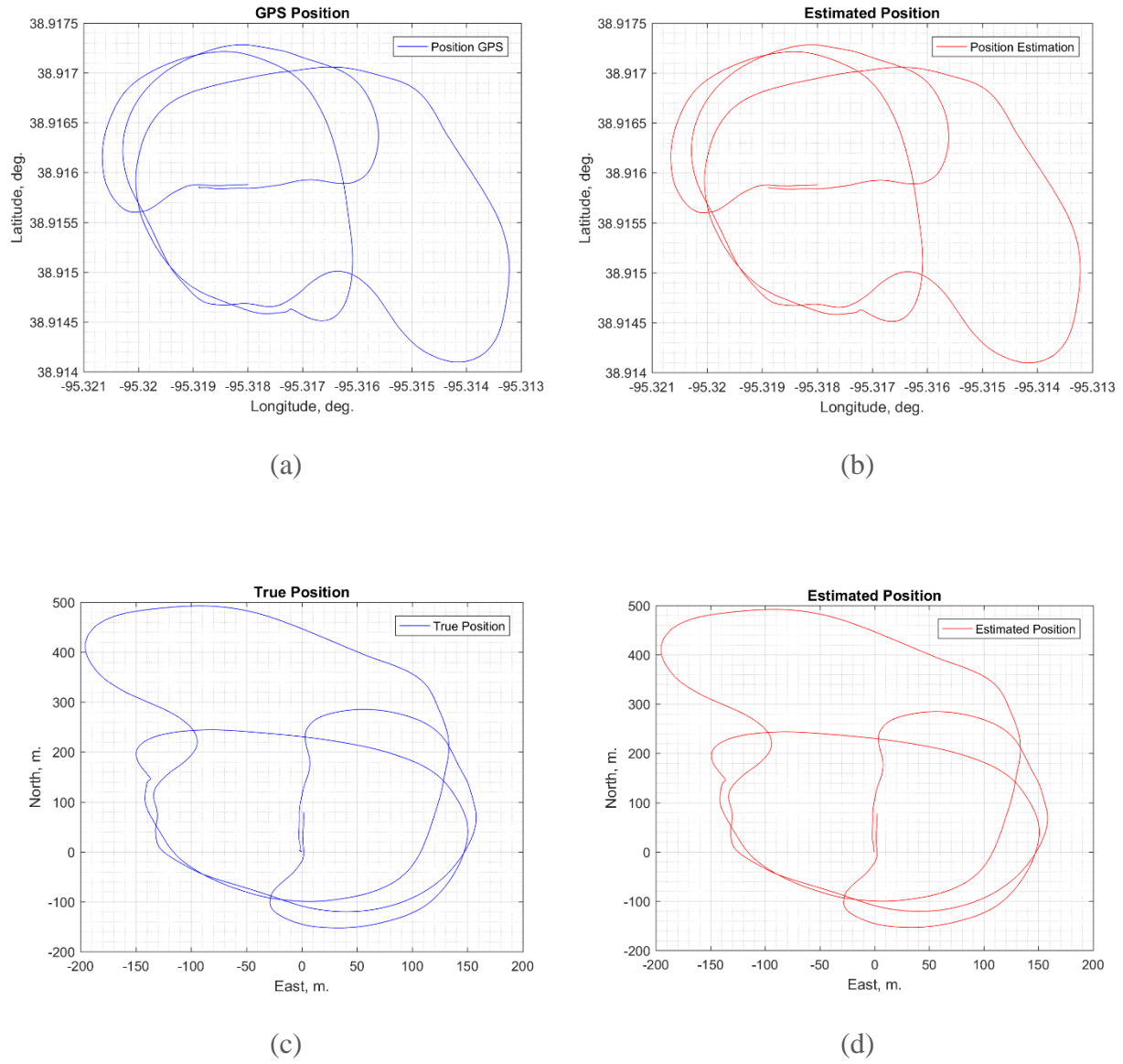
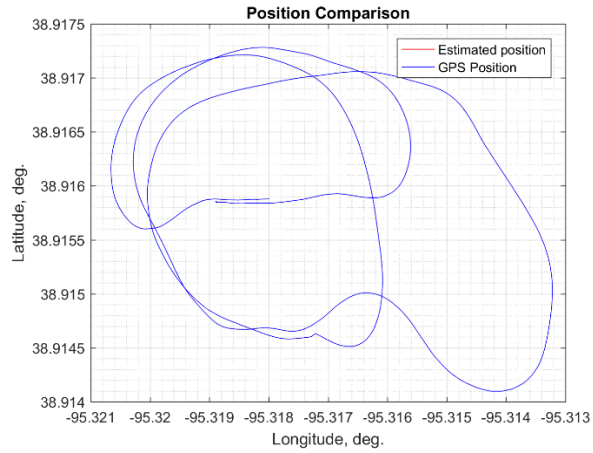
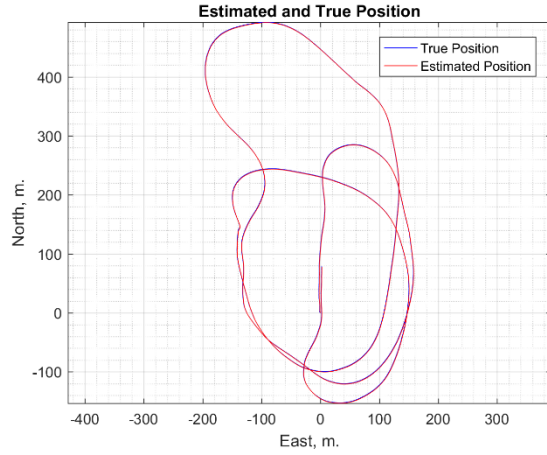


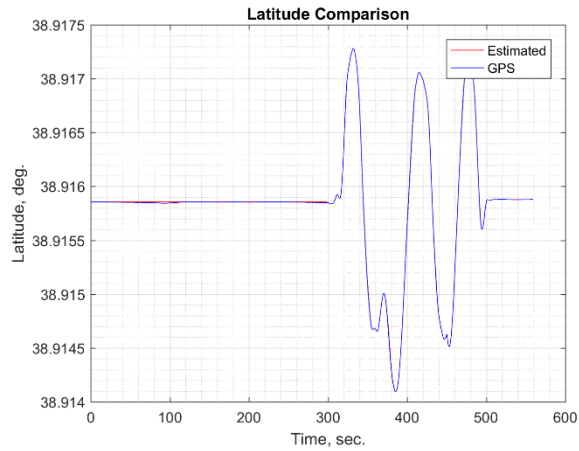
Figure A.4. True and estimated position (a) GPS position in Geodetic coordinate system, (b) Estimated position in geodetic coordinate system, (c) True position in NED (Local) coordinate system and (d) Estimated position in NED (Local) coordinate system



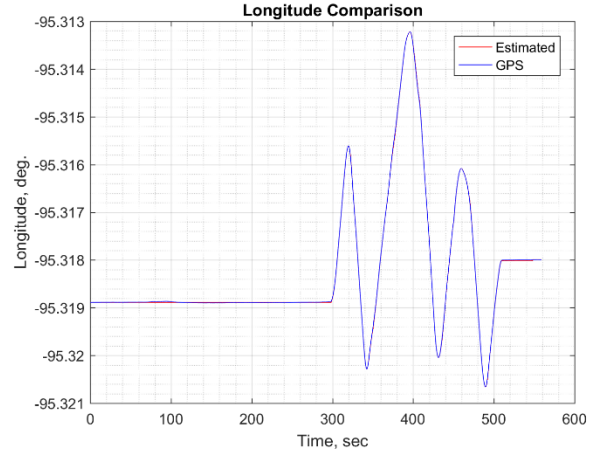
(a)



(b)

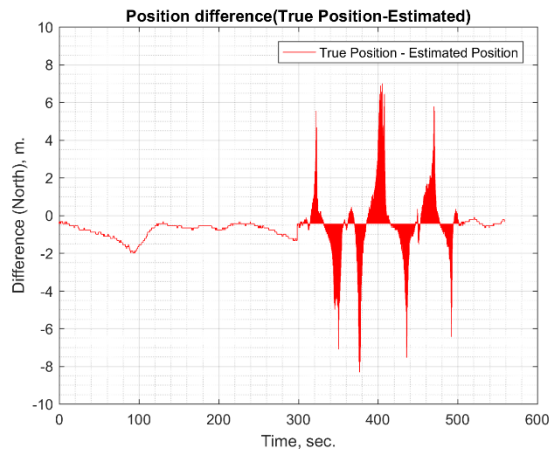


(c)

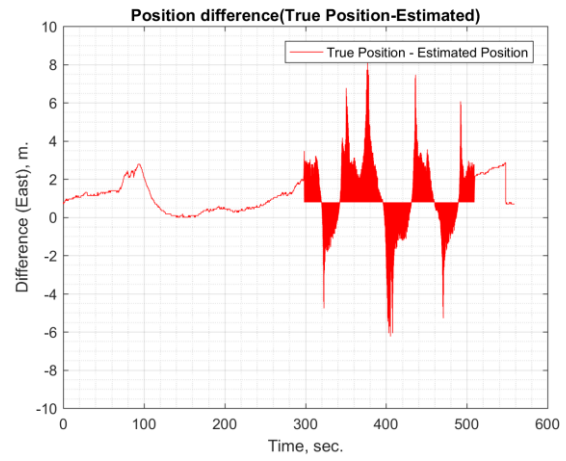


(d)

Figure A.5. True and estimated position comparison (a) GPS position and estimated position comparison in Geodetic coordinate system, (b) True position and estimated position comparison in NED (Local) coordinate system, (c)) GPS position and estimated position latitude comparison and (d) GPS position and estimated position longitude comparison



(a)



(b)

Figure A.6. Position difference with respect to time (a) Position difference in North, (b) Position difference in East

DATA SET #3

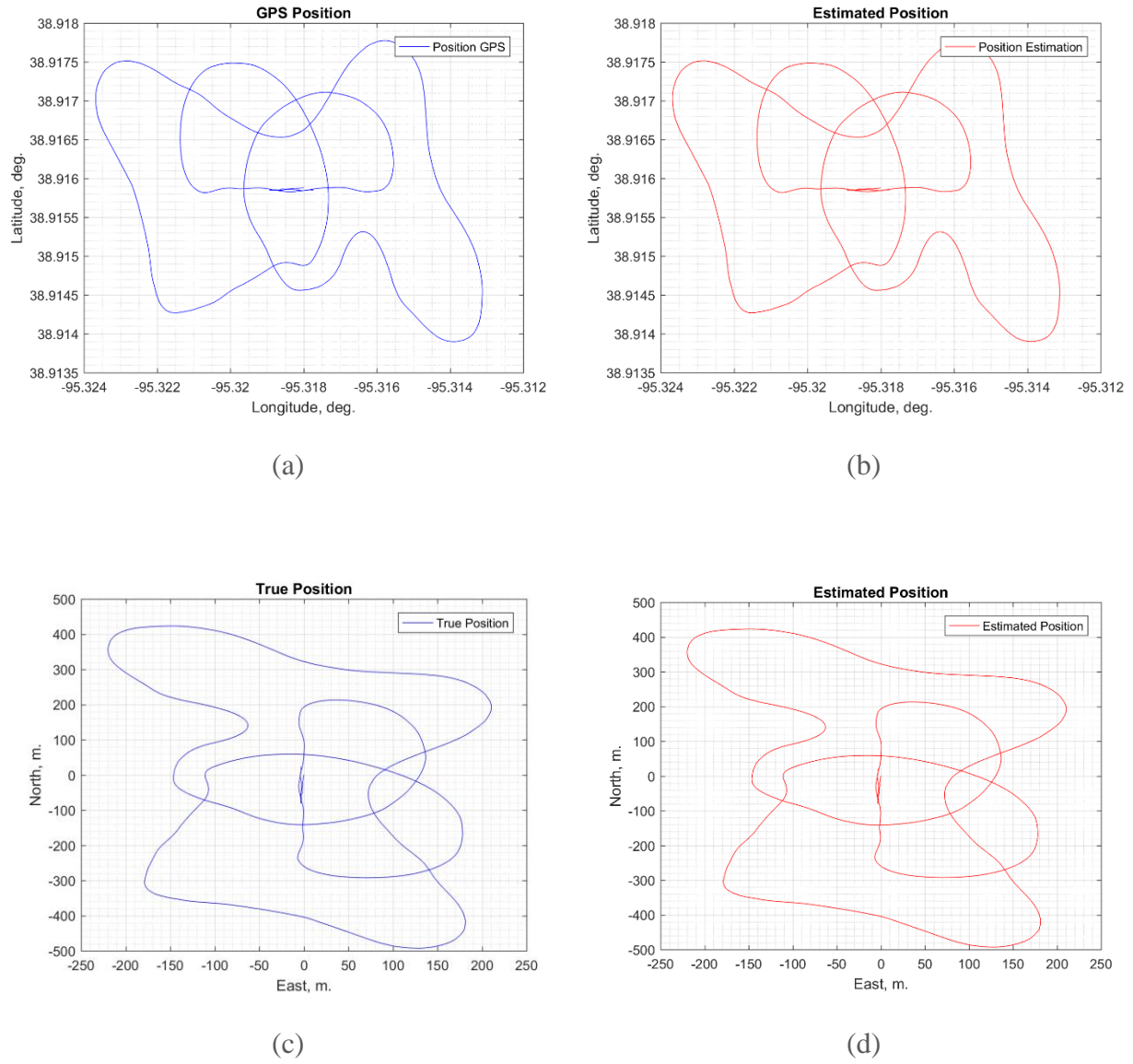
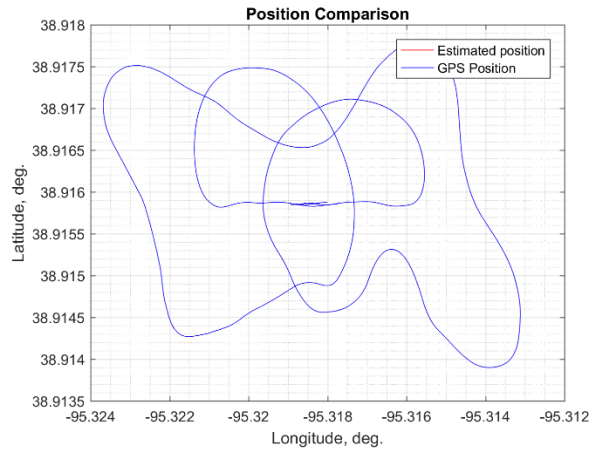
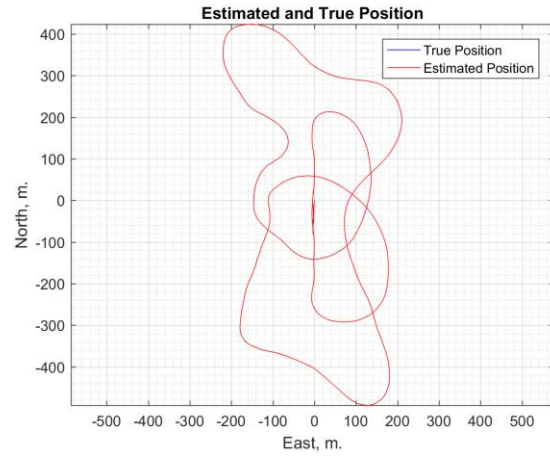


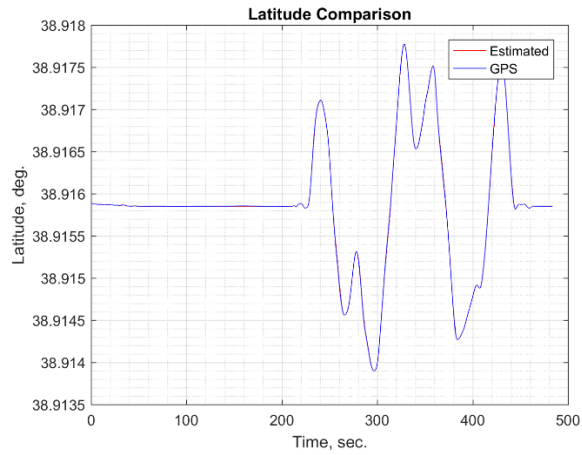
Figure A.7. True and estimated position (a) GPS position in Geodetic coordinate system, (b) Estimated position in geodetic coordinate system, (c) True position in NED (Local) coordinate system and (d) Estimated position in NED (Local) coordinate system



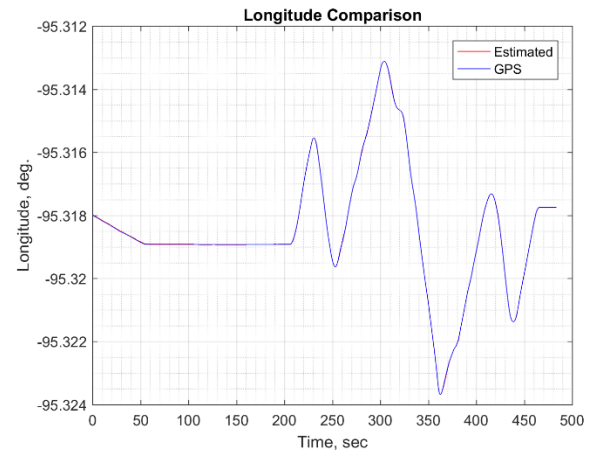
(a)



(b)

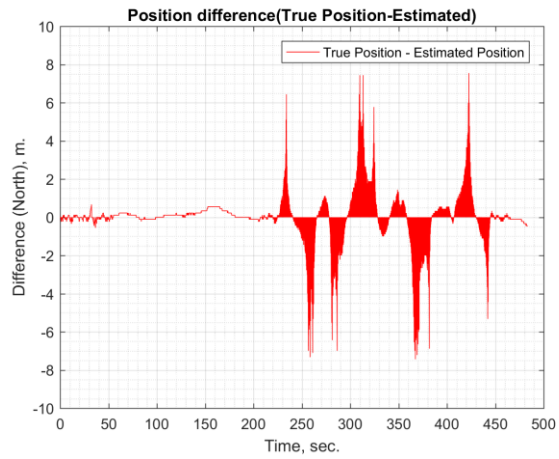


(c)

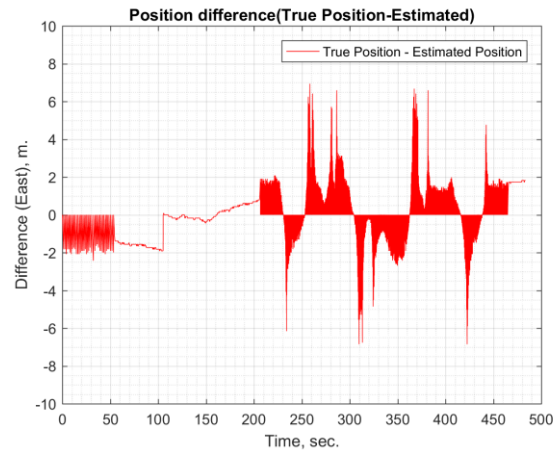


(d)

Figure A.8. True and estimated position comparison (a) GPS position and estimated position comparison in Geodetic coordinate system, (b) True position and estimated position comparison in NED (Local) coordinate system, (c)) GPS position and estimated position latitude comparison and (d) GPS position and estimated position longitude comparison



(a)



(b)

Figure A.9. Position difference with respect to time (a) Position difference in North, (b) Position difference in East

DATA SET #4

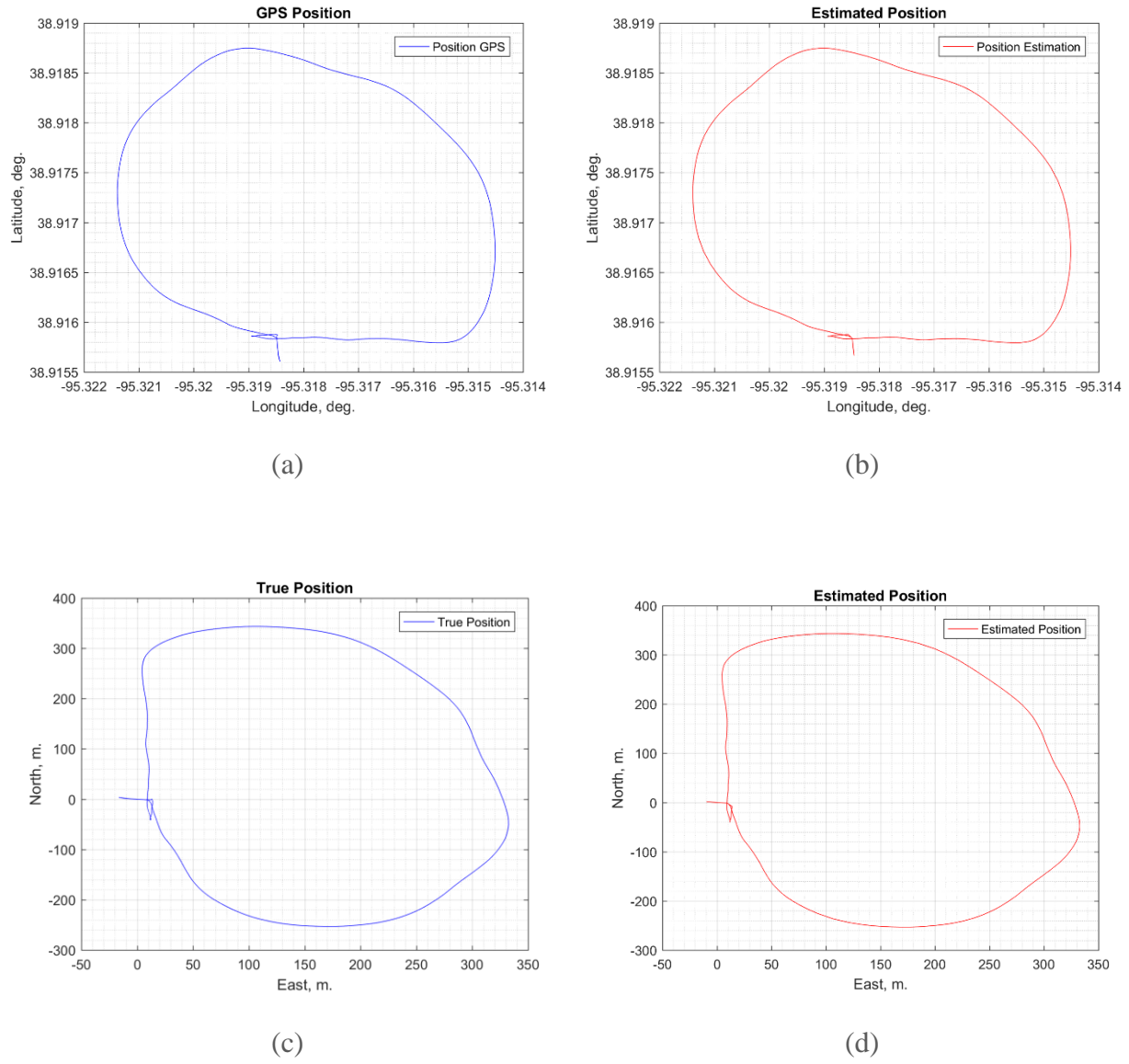
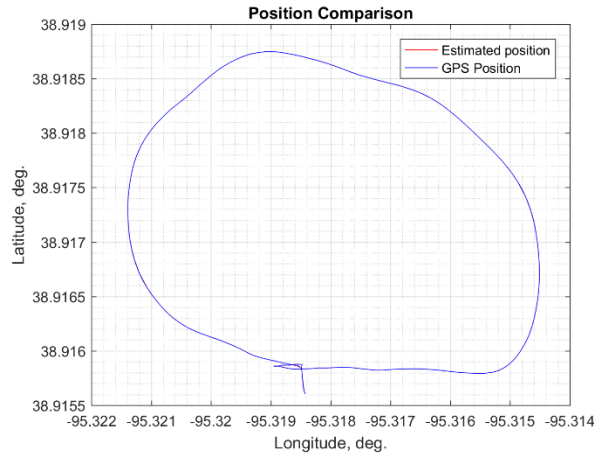
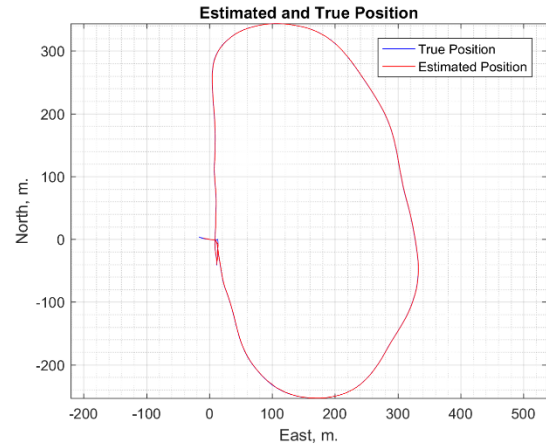


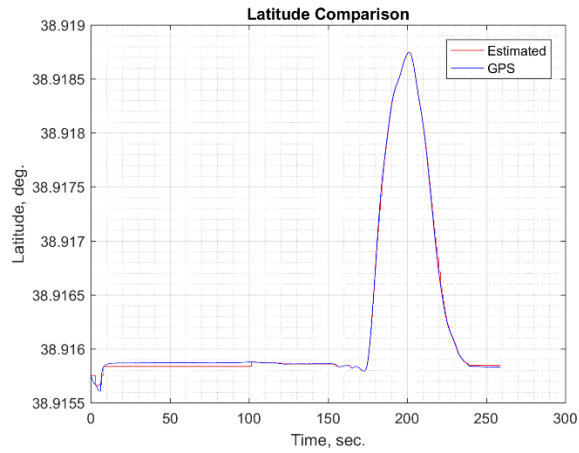
Figure A.10. True and estimated position (a) GPS position in Geodetic coordinate system, (b) Estimated position in geodetic coordinate system, (c) True position in NED (Local) coordinate system and (d) Estimated position in NED (Local) coordinate system



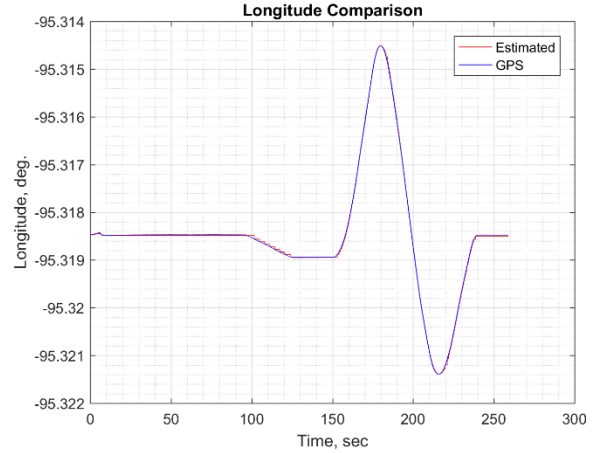
(a)



(b)



(c)



(d)

Figure A.11. True and estimated position comparison (a) GPS position and estimated position comparison in Geodetic coordinate system, (b) True position and estimated position comparison in NED (Local) coordinate system, (c)) GPS position and estimated position latitude comparison and (d) GPS position and estimated position longitude comparison



(a)



(b)

Figure A.12. Position difference with respect to time (a) Position difference in North, (b) Position difference in East

DATA SET #5

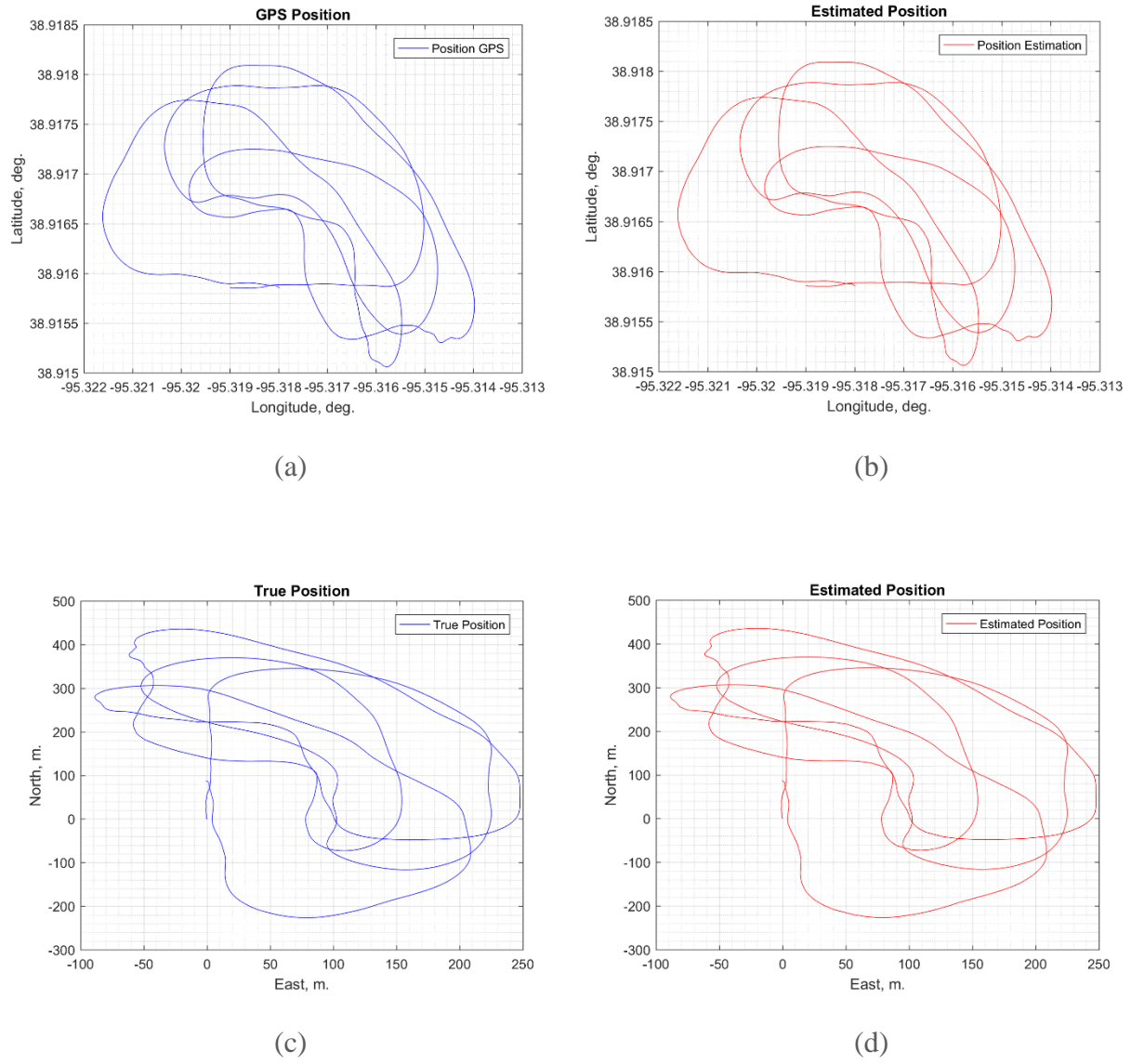
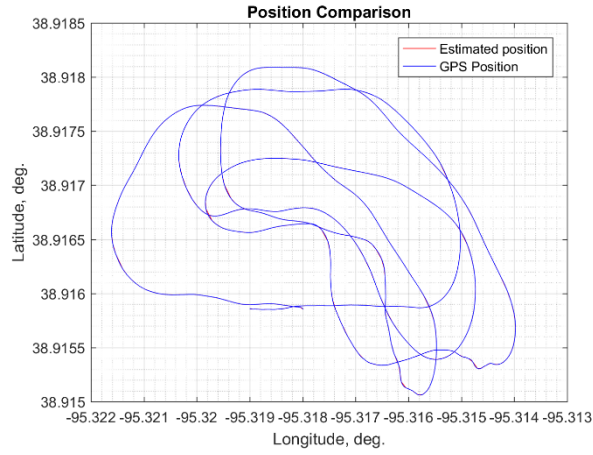


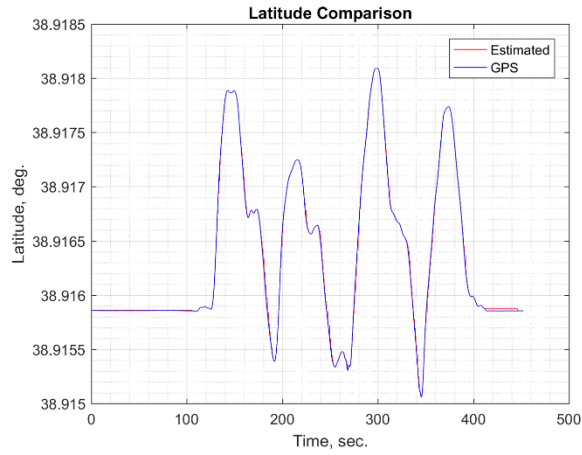
Figure A.13. True and estimated position (a) GPS position in Geodetic coordinate system, (b) Estimated position in geodetic coordinate system, (c) True position in NED (Local) coordinate system and (d) Estimated position in NED (Local) coordinate system



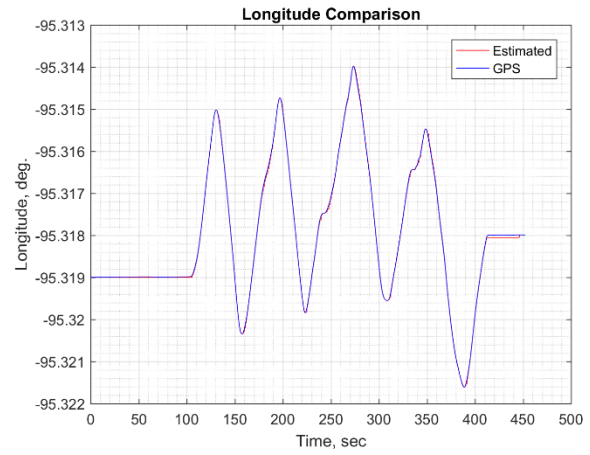
(a)



(b)

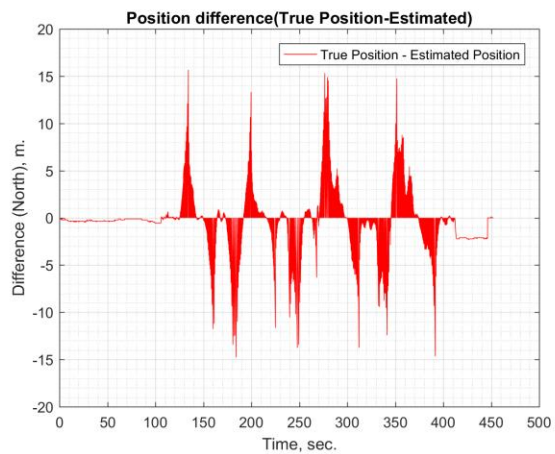


(c)

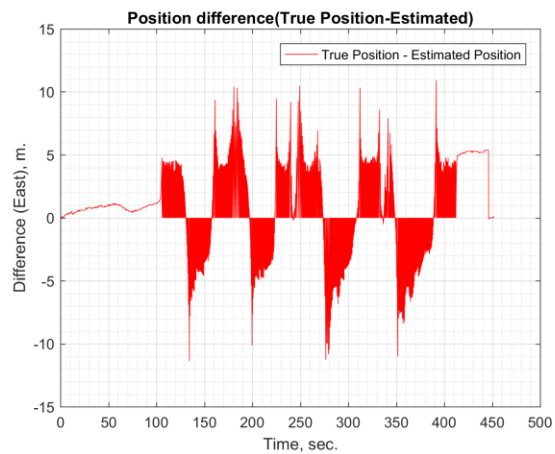


(d)

Figure A.14. True and estimated position comparison (a) GPS position and estimated position comparison in Geodetic coordinate system, (b) True position and estimated position comparison in NED (Local) coordinate system, (c)) GPS position and estimated position latitude comparison and (d) GPS position and estimated position longitude comparison



(a)



(b)

Figure A.15. Position difference with respect to time (a) Position difference in North, (b) Position difference in East

DATA SET #6

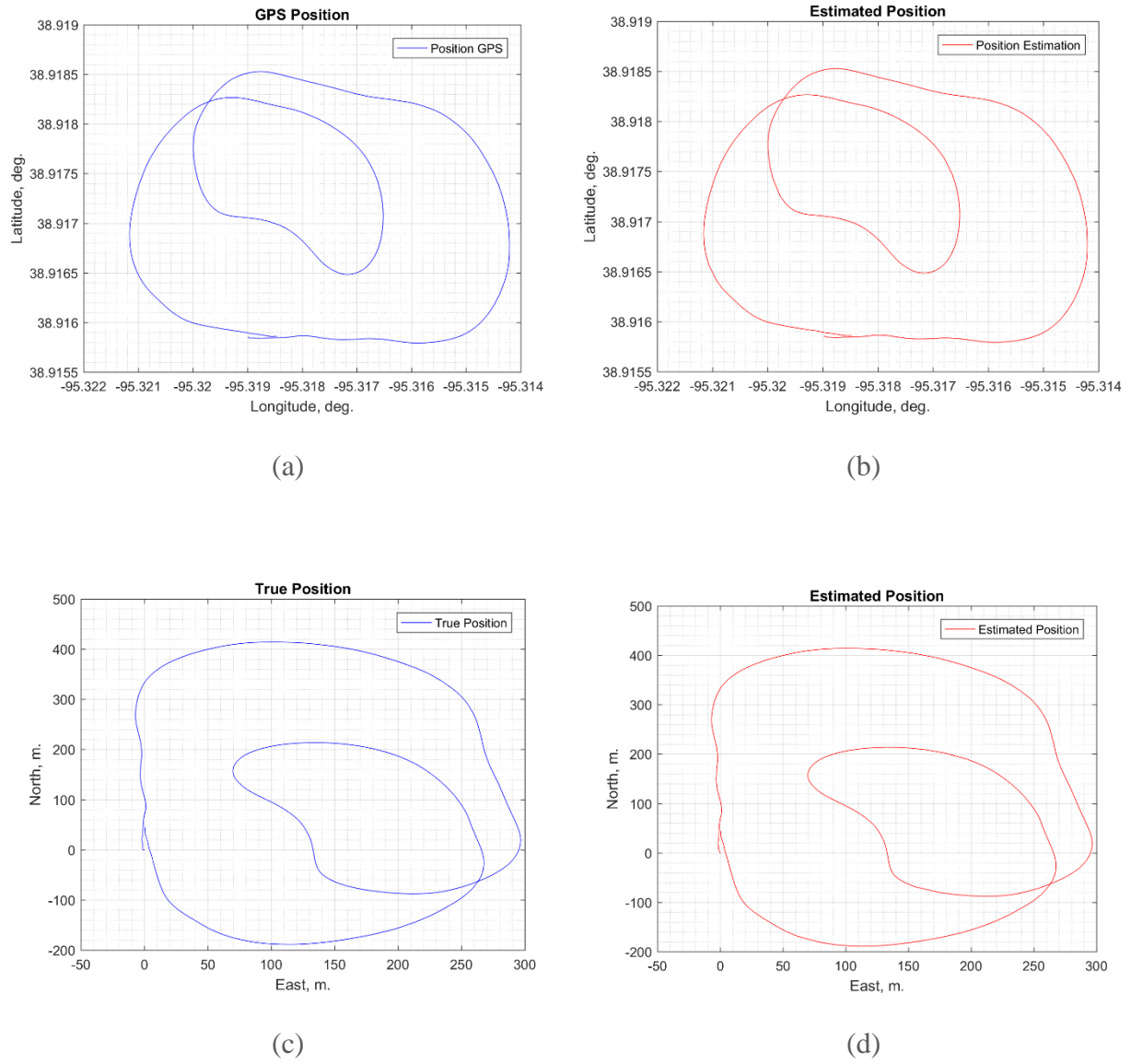
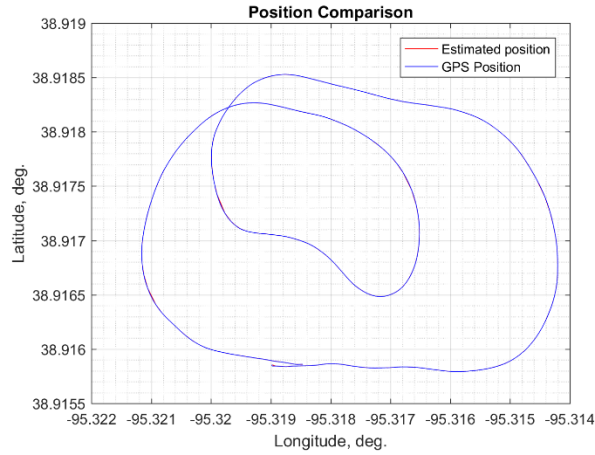
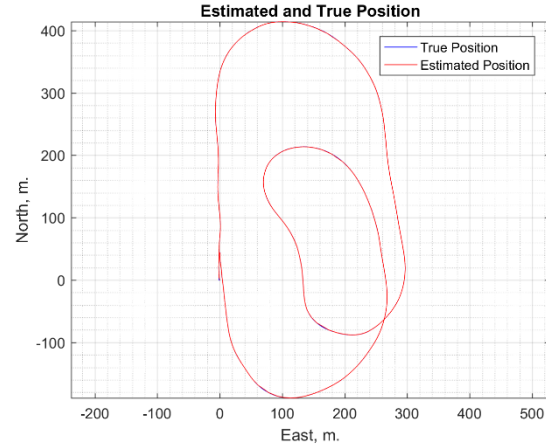


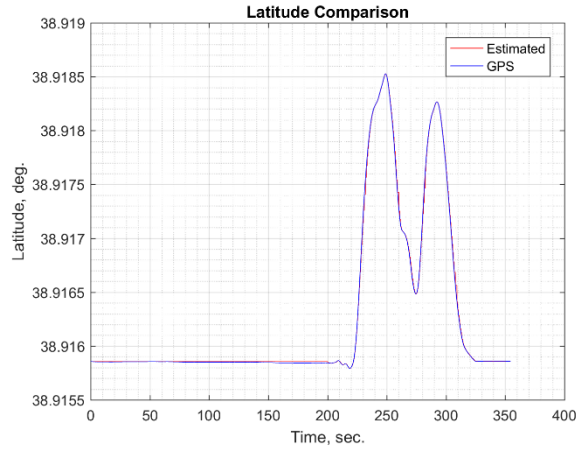
Figure A.16. True and estimated position (a) GPS position in Geodetic coordinate system, (b) Estimated position in geodetic coordinate system, (c) True position in NED (Local) coordinate system and (d) Estimated position in NED (Local) coordinate system



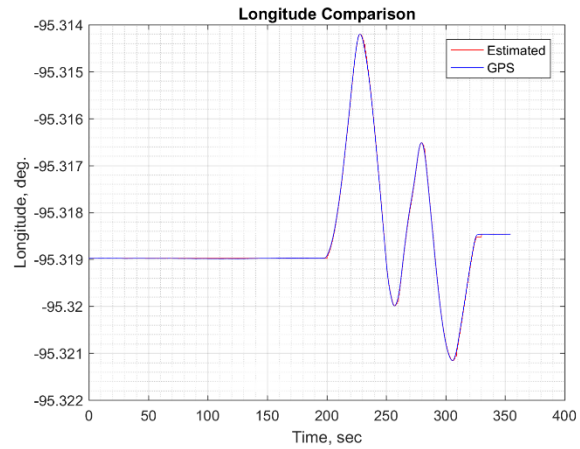
(a)



(b)

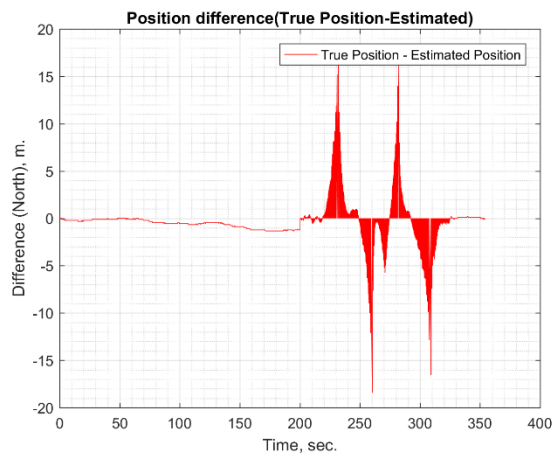


(c)

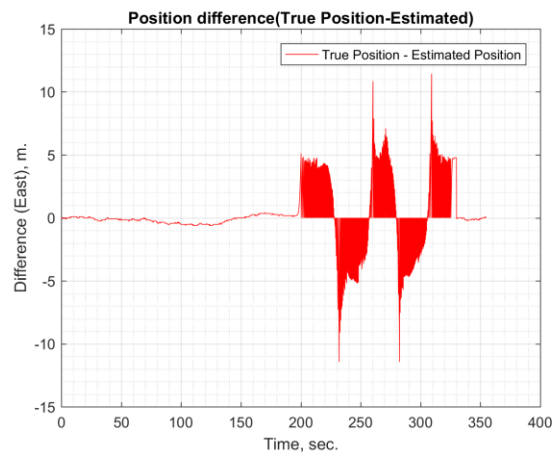


(d)

Figure A.17. True and estimated position comparison (a) GPS position and estimated position comparison in Geodetic coordinate system, (b) True position and estimated position comparison in NED (Local) coordinate system, (c)) GPS position and estimated position latitude comparison and (d) GPS position and estimated position longitude comparison



(a)



(b)

Figure A.18. Position difference with respect to time (a) Position difference in North, (b) Position difference in East

DATA SET #7

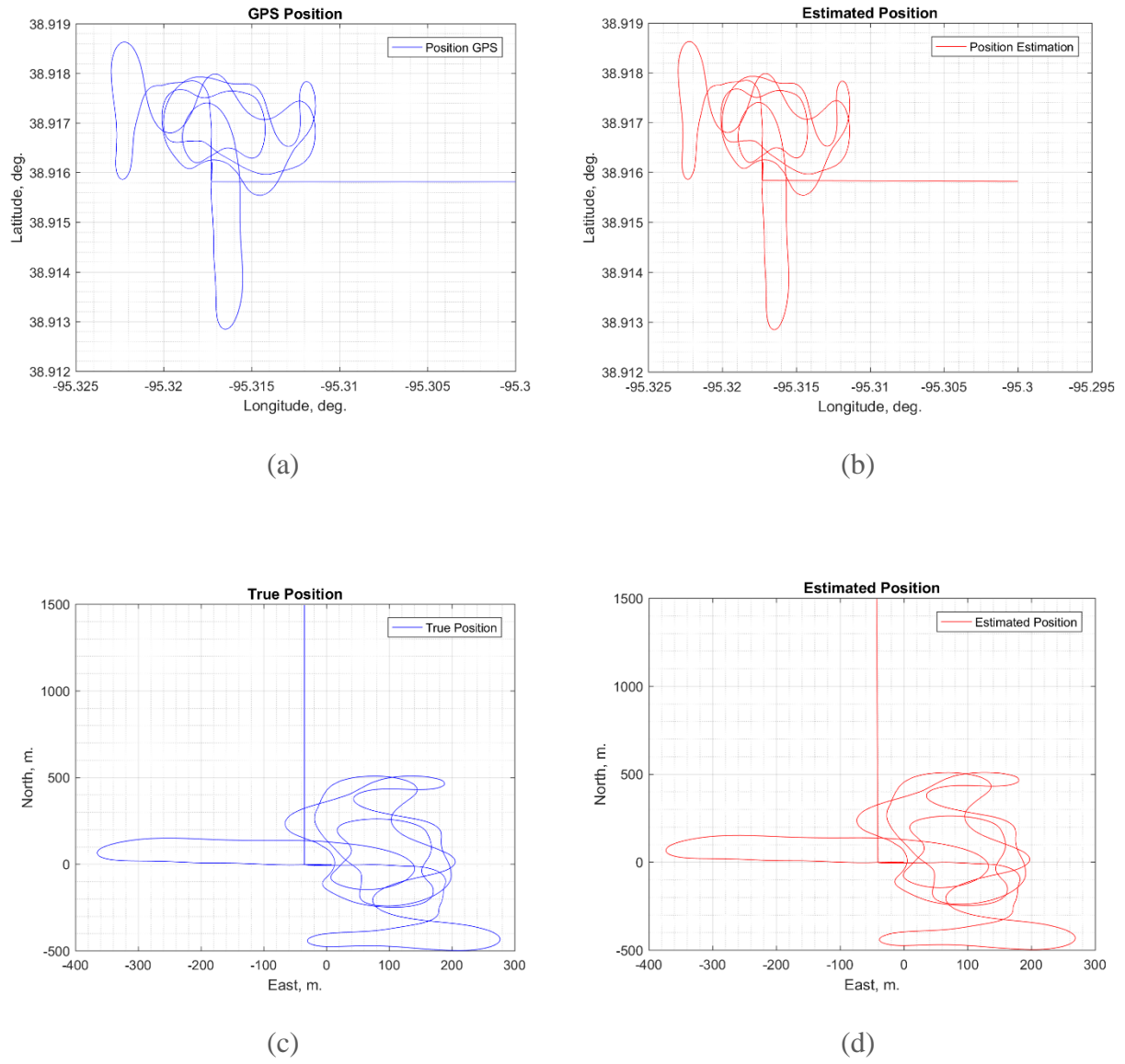
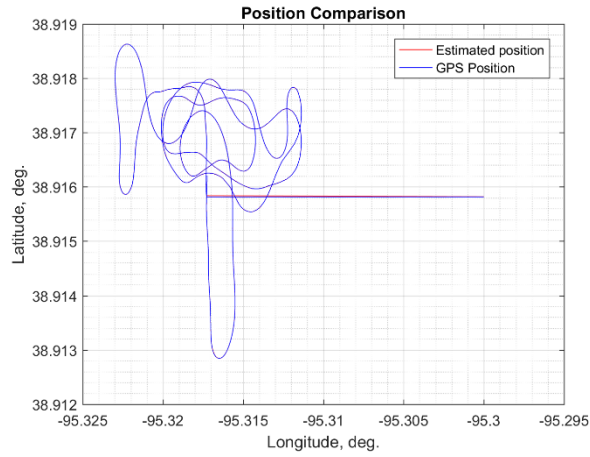
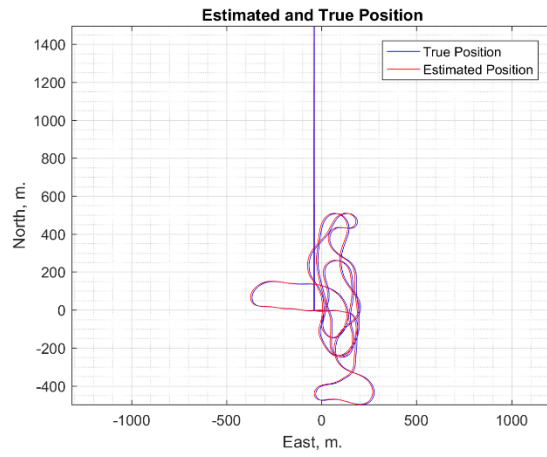


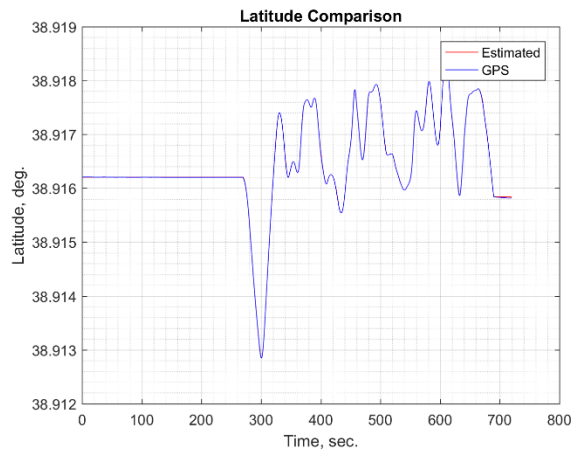
Figure A.19. True and estimated position (a) GPS position in Geodetic coordinate system, (b) Estimated position in geodetic coordinate system, (c) True position in NED (Local) coordinate system and (d) Estimated position in NED (Local) coordinate system



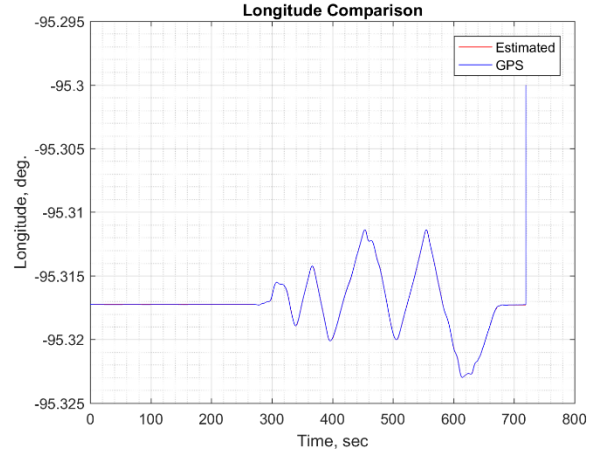
(a)



(b)

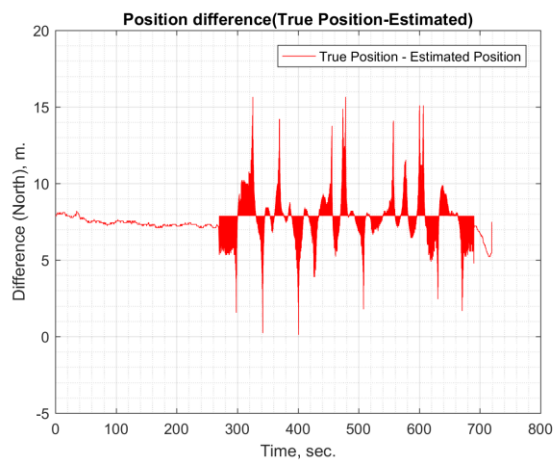


(c)

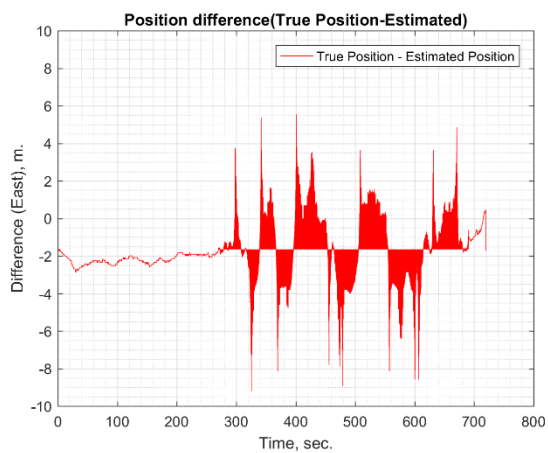


(d)

Figure A.20. True and estimated position comparison (a) GPS position and estimated position comparison in Geodetic coordinate system, (b) True position and estimated position comparison in NED (Local) coordinate system, (c)) GPS position and estimated position latitude comparison and (d) GPS position and estimated position longitude comparison



(a)



(b)

Figure A.21. Position difference with respect to time (a) Position difference in North, (b) Position difference in East

DATA SET #8

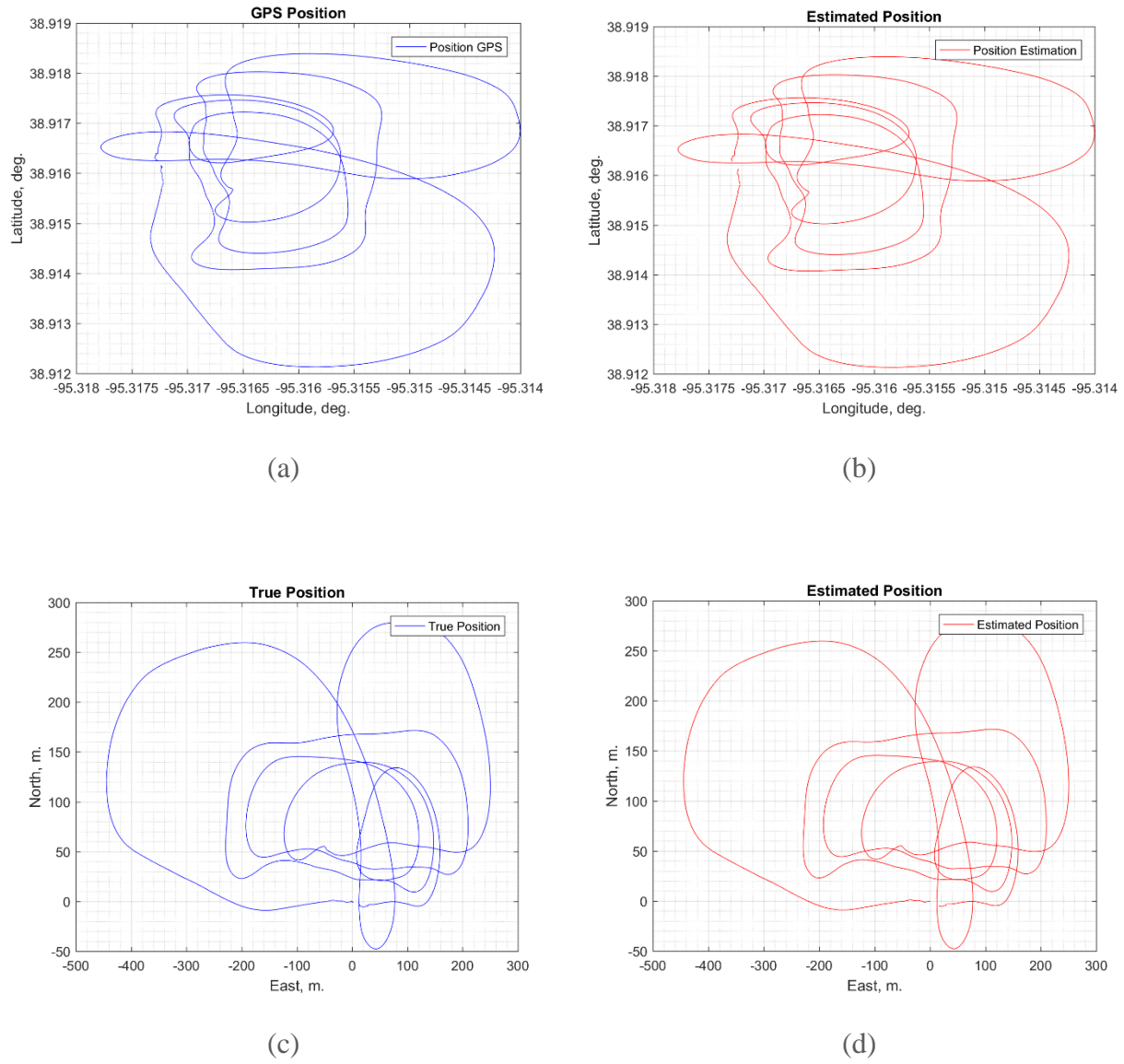
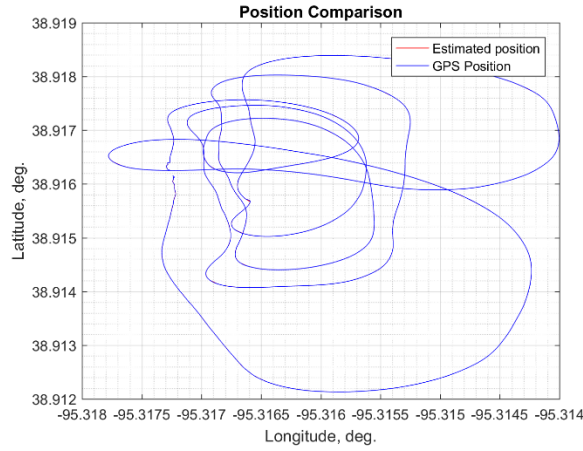
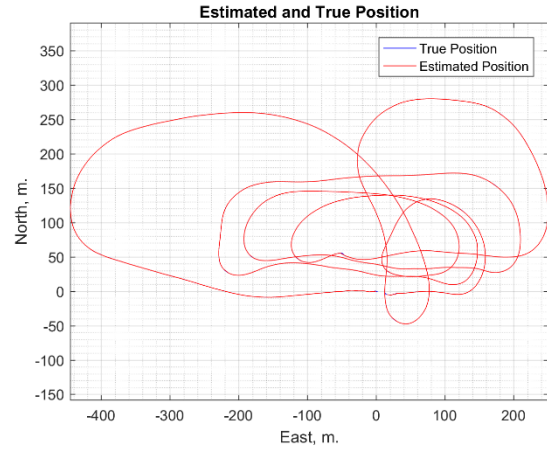


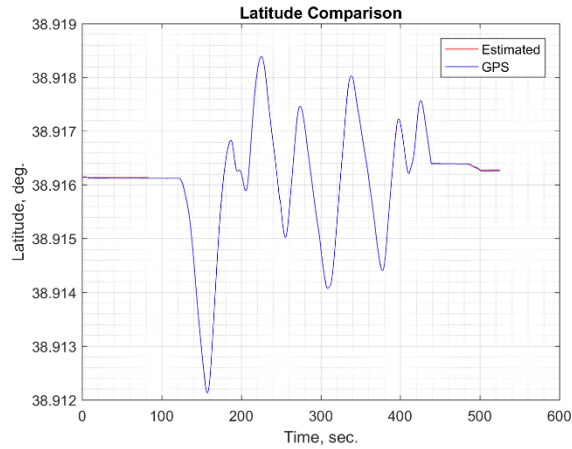
Figure A.22. True and estimated position (a) GPS position in Geodetic coordinate system, (b) Estimated position in geodetic coordinate system, (c) True position in NED (Local) coordinate system and (d) Estimated position in NED (Local) coordinate system



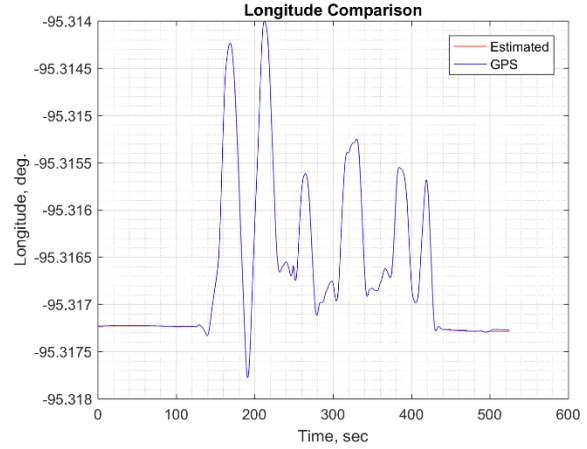
(a)



(b)

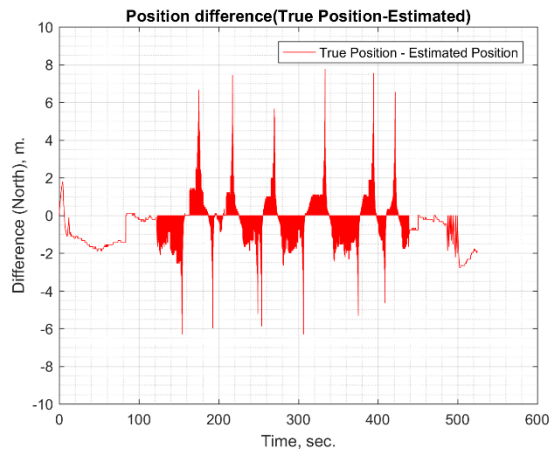


(c)

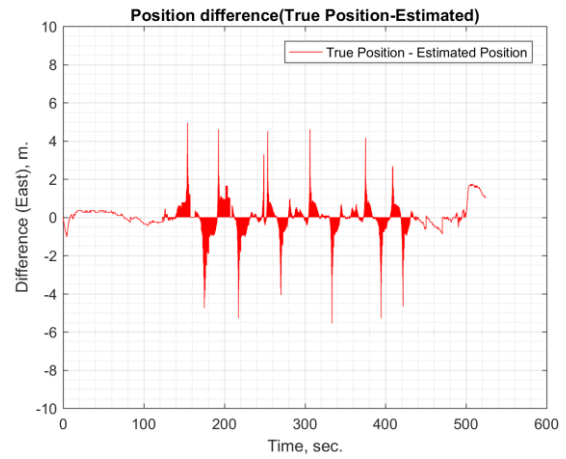


(d)

Figure A.23. True and estimated position comparison (a) GPS position and estimated position comparison in Geodetic coordinate system, (b) True position and estimated position comparison in NED (Local) coordinate system, (c)) GPS position and estimated position latitude comparison and (d) GPS position and estimated position longitude comparison



(a)



(b)

Figure A.24. Position difference with respect to time (a) Position difference in North, (b) Position difference in East

DATA SET #9

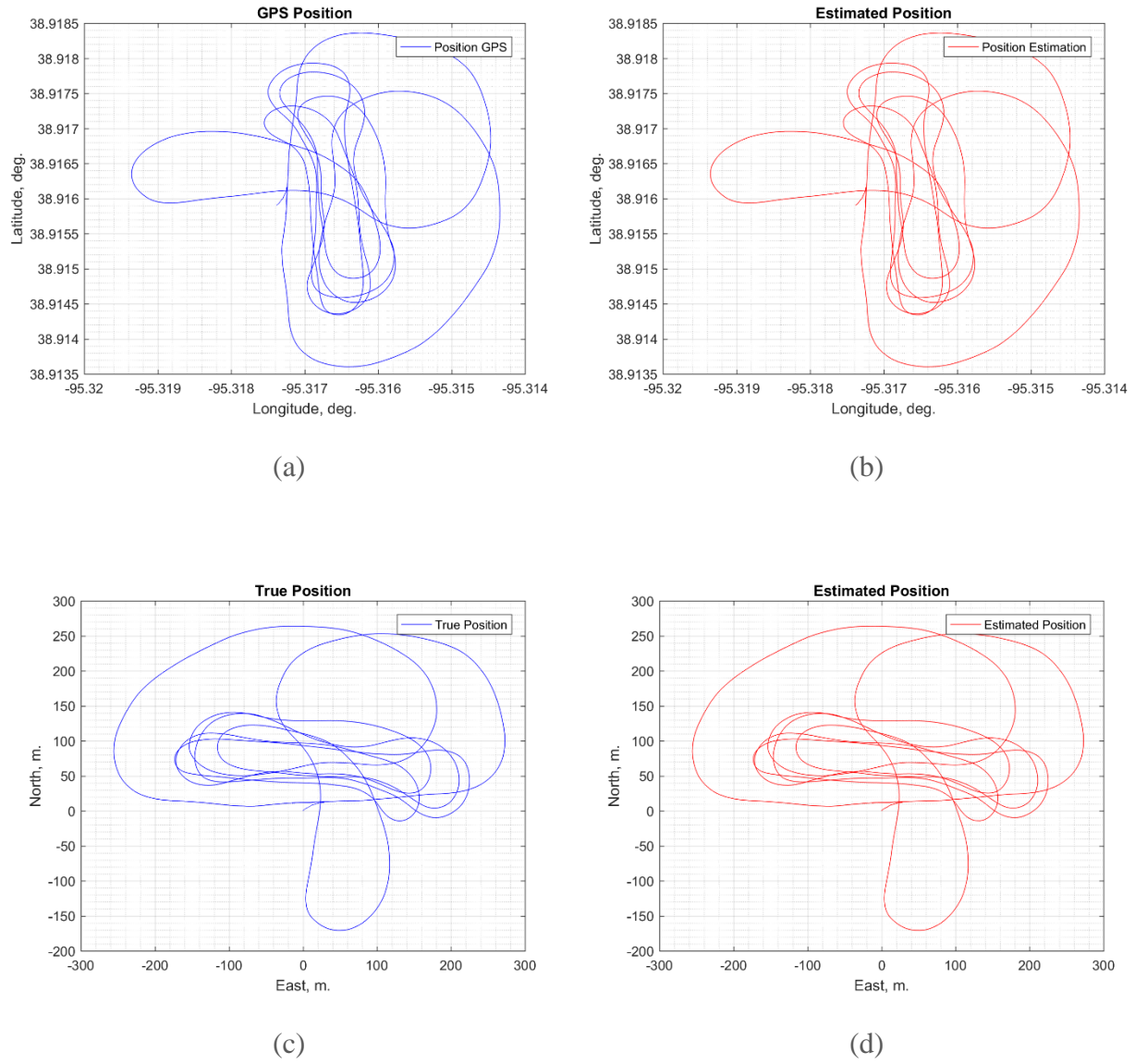
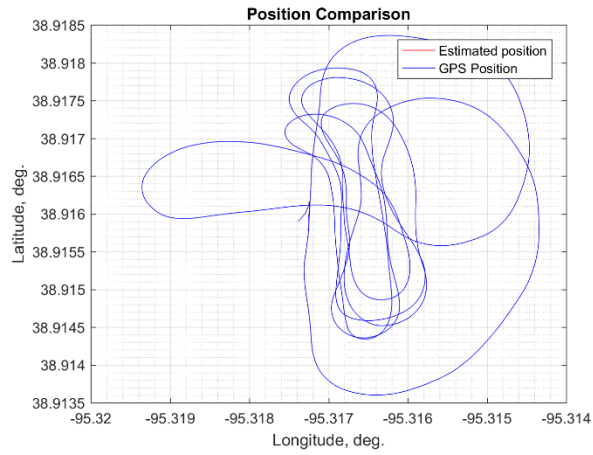
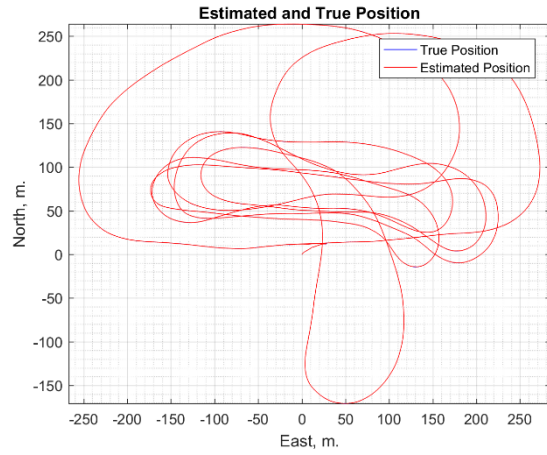


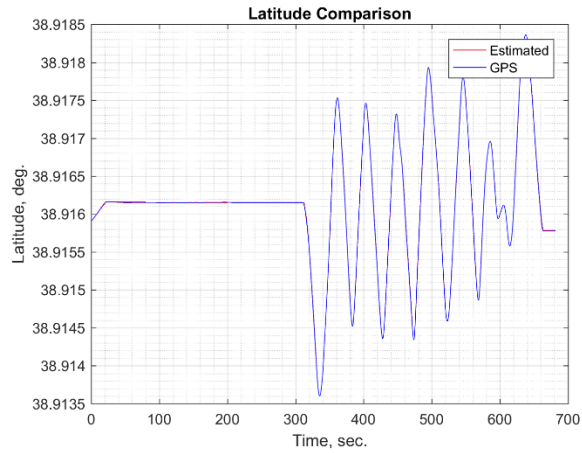
Figure A.25. True and estimated position (a) GPS position in Geodetic coordinate system, (b) Estimated position in geodetic coordinate system, (c) True position in NED (Local) coordinate system and (d) Estimated position in NED (Local) coordinate system



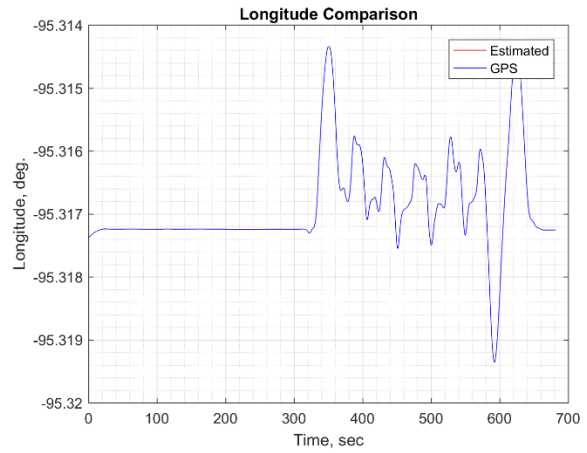
(a)



(b)

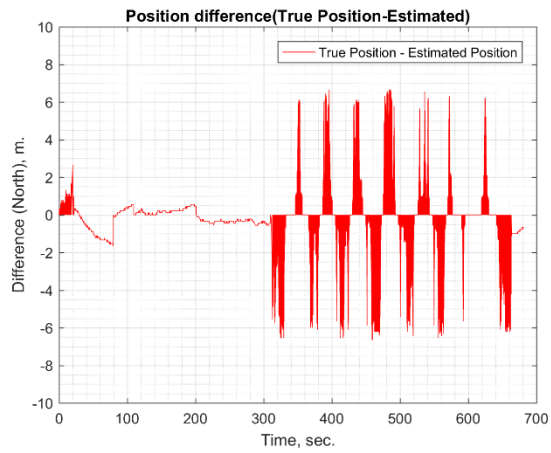


(c)

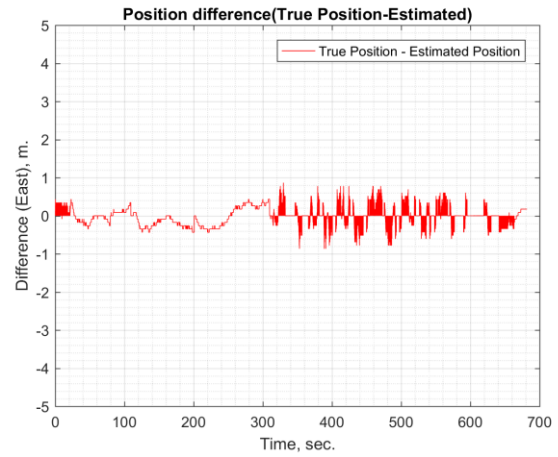


(d)

Figure A.26. True and estimated position comparison (a) GPS position and estimated position comparison in Geodetic coordinate system, (b) True position and estimated position comparison in NED (Local) coordinate system, (c)) GPS position and estimated position latitude comparison and (d) GPS position and estimated position longitude comparison



(a)



(b)

Figure A.27. Position difference with respect to time (a) Position difference in North, (b) Position difference in East

DATA SET #10

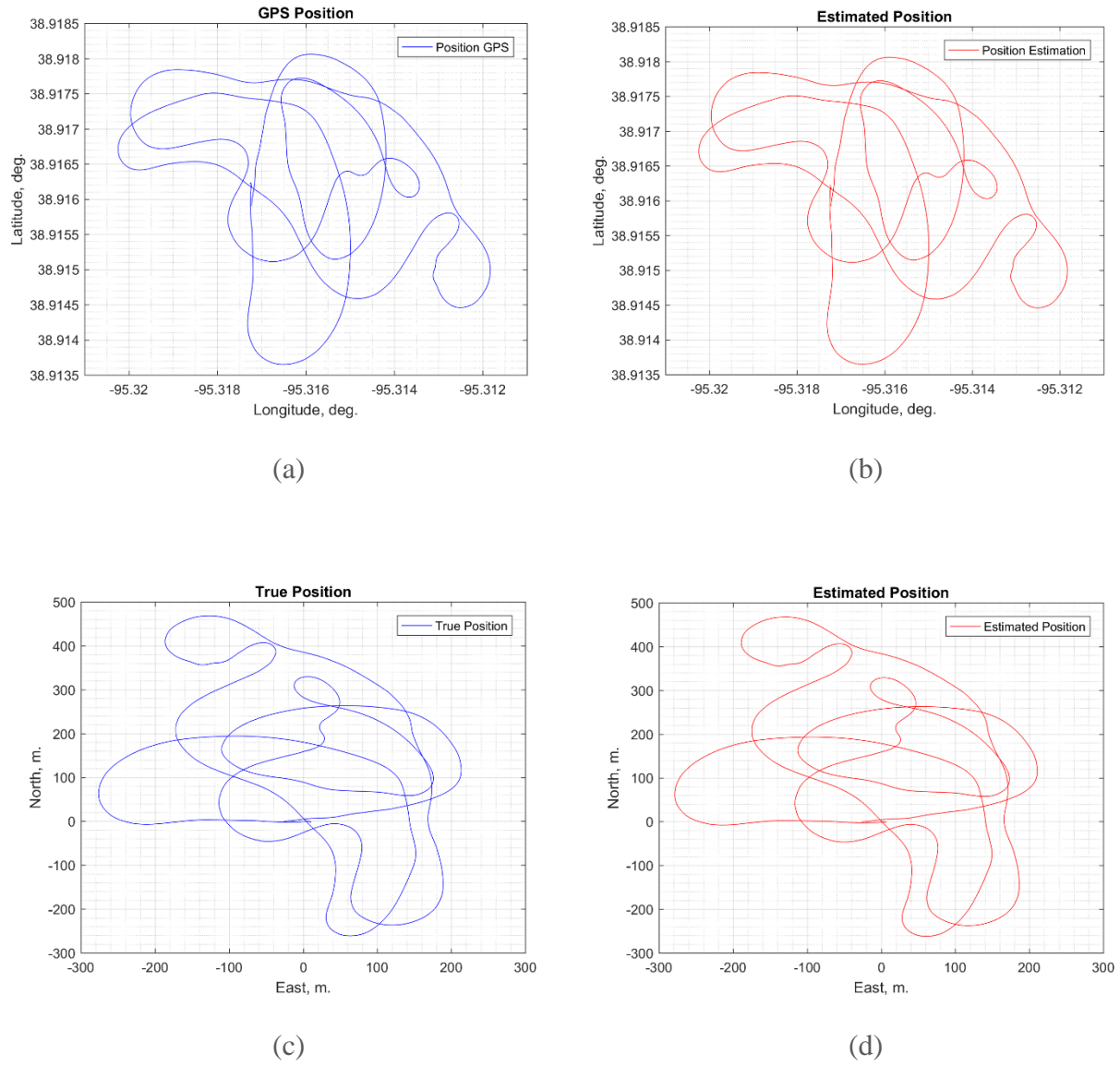
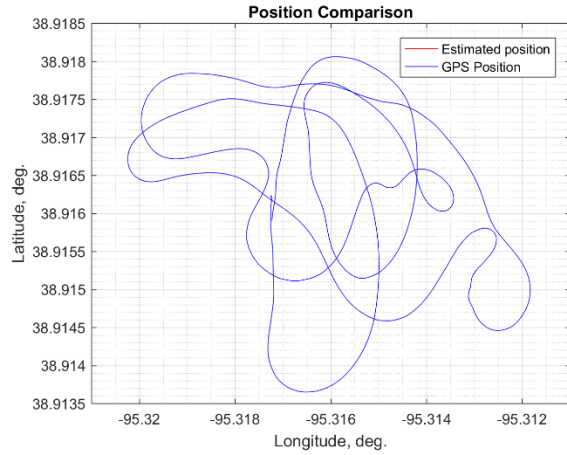
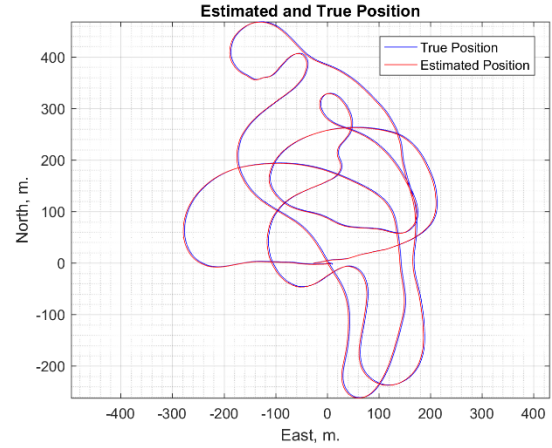


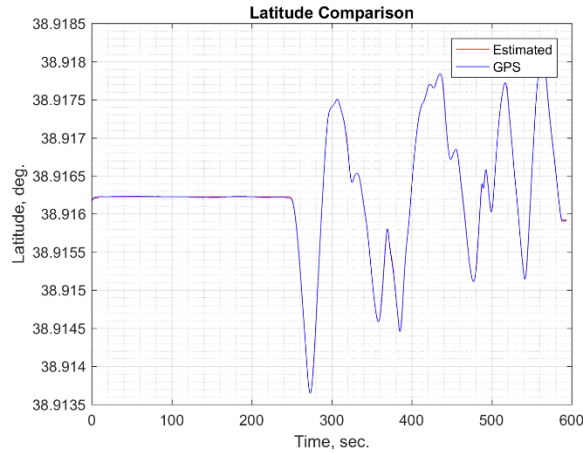
Figure A.28. True and estimated position (a) GPS position in Geodetic coordinate system, (b) Estimated position in geodetic coordinate system, (c) True position in NED (Local) coordinate system and (d) Estimated position in NED (Local) coordinate system



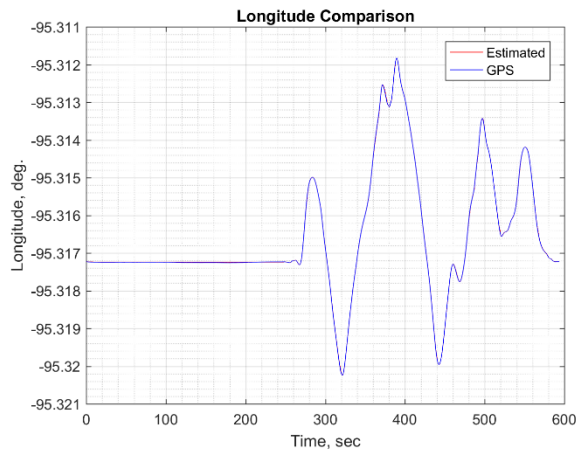
(a)



(b)



(c)

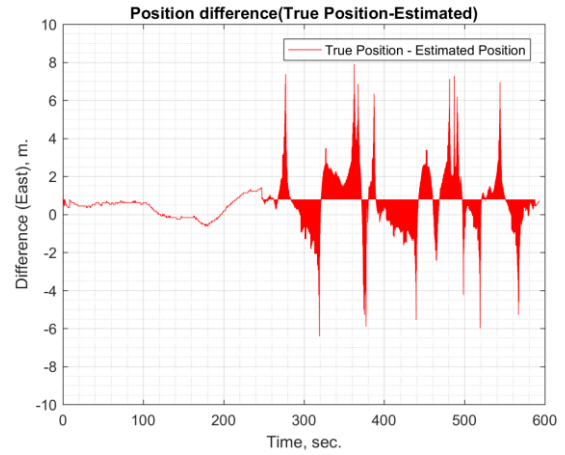


(d)

Figure A.29. True and estimated position comparison (a) GPS position and estimated position comparison in Geodetic coordinate system, (b) True position and estimated position comparison in NED (Local) coordinate system, (c)) GPS position and estimated position latitude comparison and (d) GPS position and estimated position longitude comparison



(a)



(b)

Figure A.30. Position difference with respect to time (a) Position difference in North, (b) Position difference in East

Appendix B. Correlation of error with time of day and day

DATA SET#1

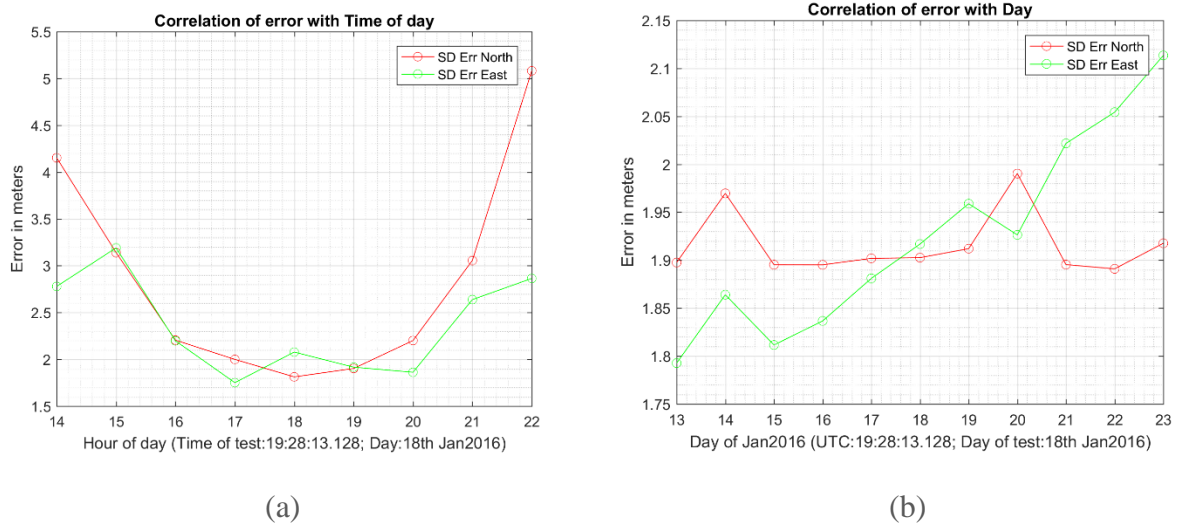


Figure B.1. Correlation of error (a) Correlation of error with time of day, (b) Correlation of error with day

DATA SET#2

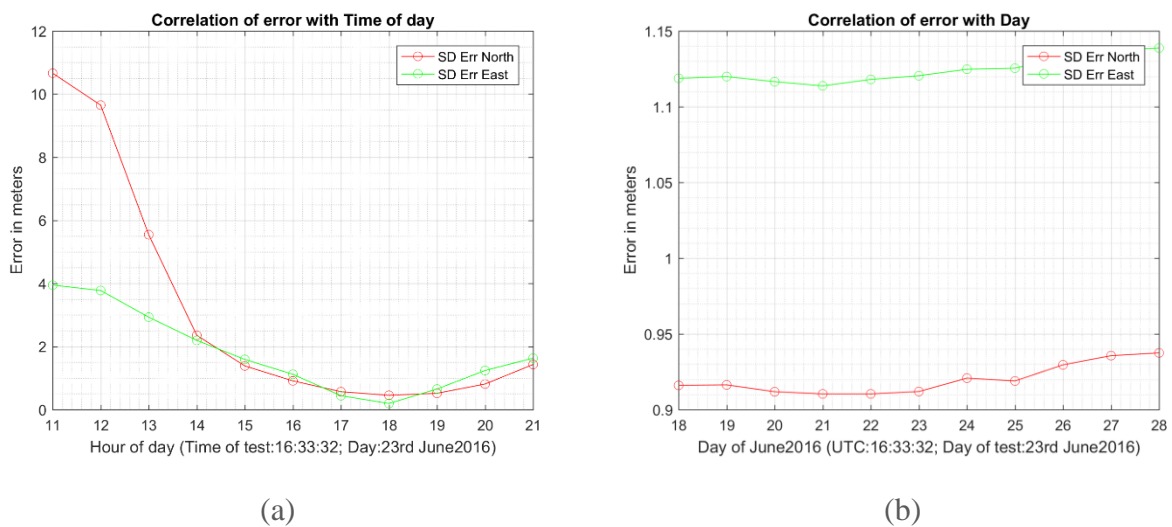
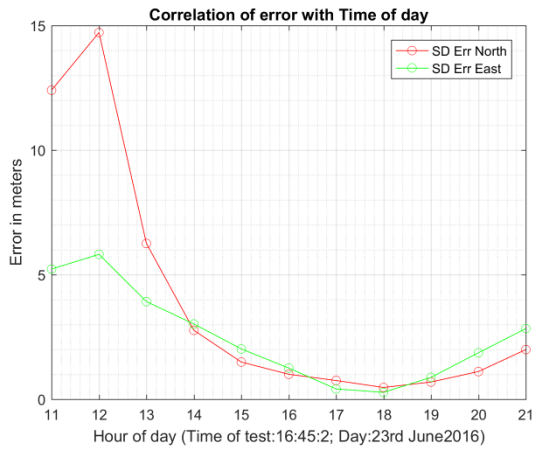
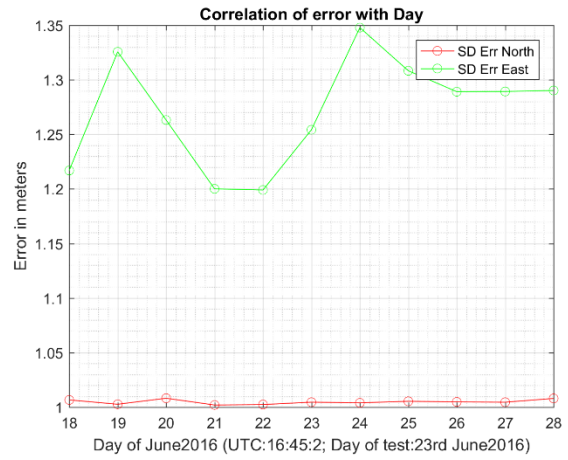


Figure B.2. Correlation of error (a) Correlation of error with time of day, (b) Correlation of error with day

DATA SET#3



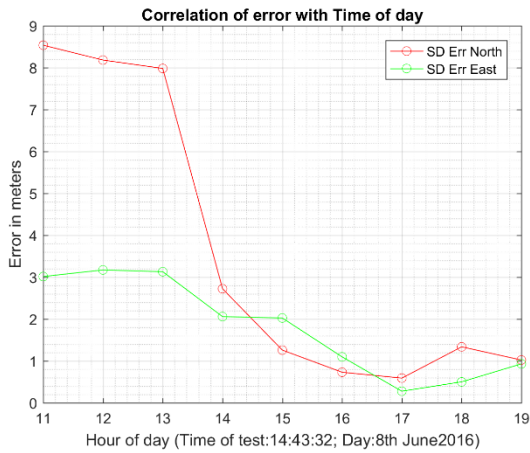
(a)



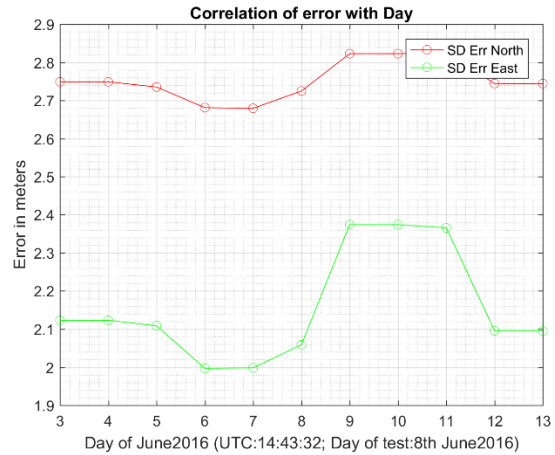
(b)

Figure B.3. Correlation of error (a) Correlation of error with time of day, (b) Correlation of error with day

DATA SET#4



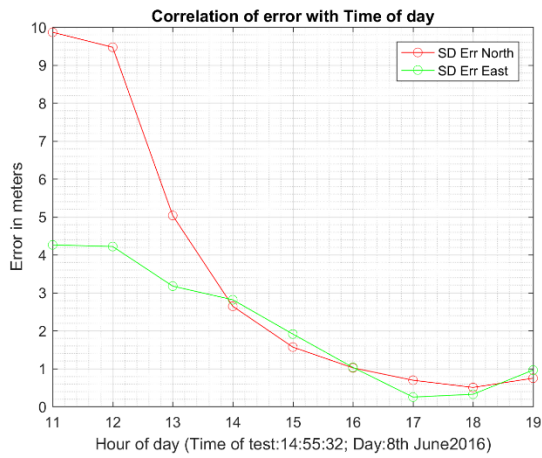
(a)



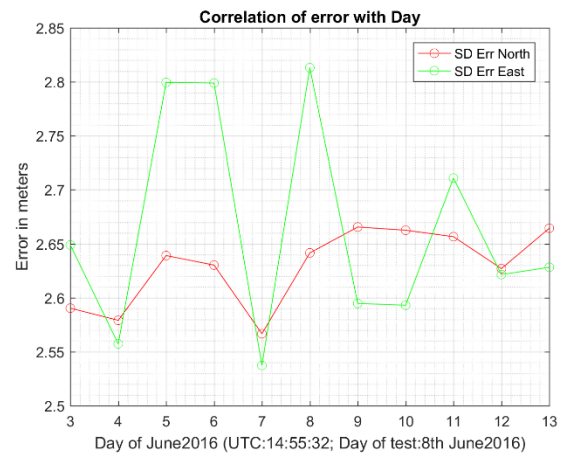
(b)

Figure B.4. Correlation of error (a) Correlation of error with time of day, (b) Correlation of error with day

DATA SET#5



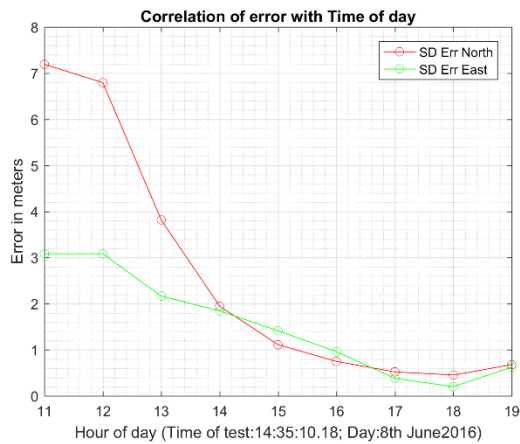
(a)



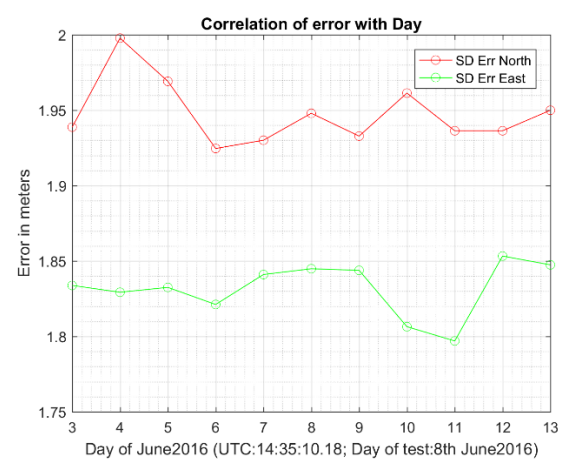
(b)

Figure B.5. Correlation of error (a) Correlation of error with time of day, (b) Correlation of error with day

DATA SET#6



(a)

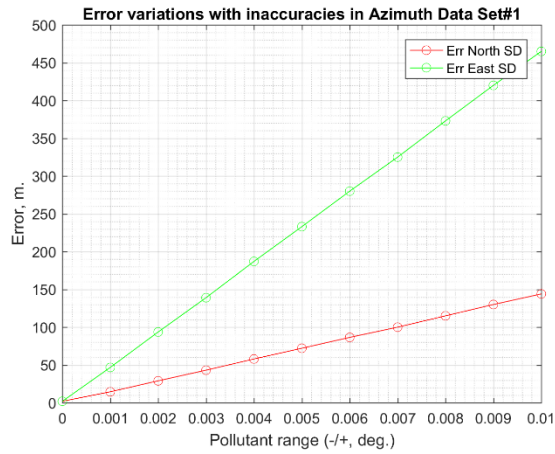


(b)

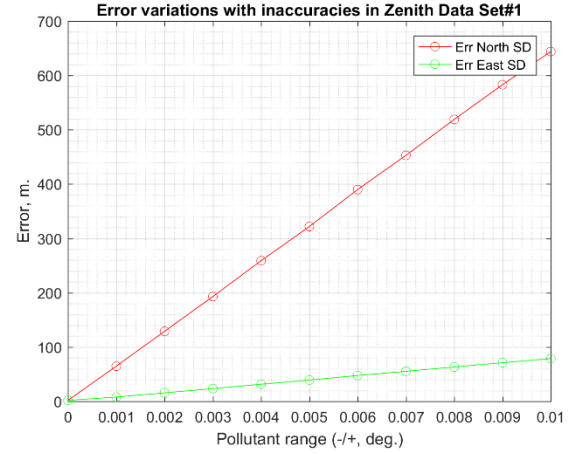
Figure B.6. Correlation of error (a) Correlation of error with time of day, (b) Correlation of error with day

Appendix C. Sensitivity of error caused by sensors

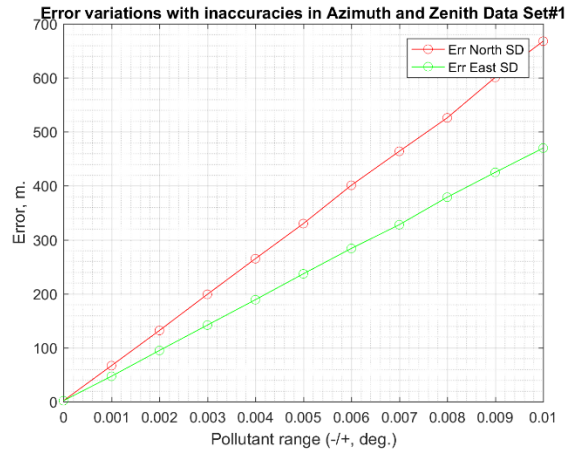
DATA SET #1



(a)



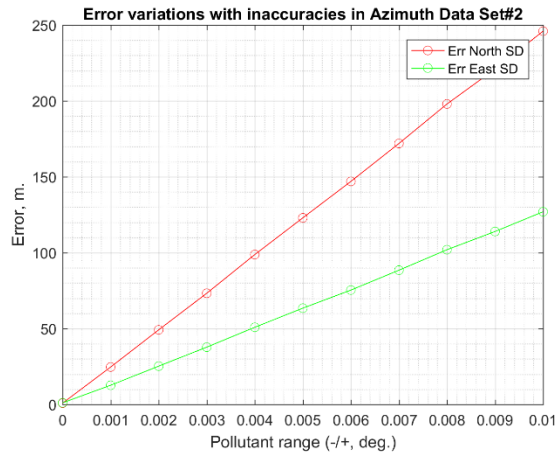
(b)



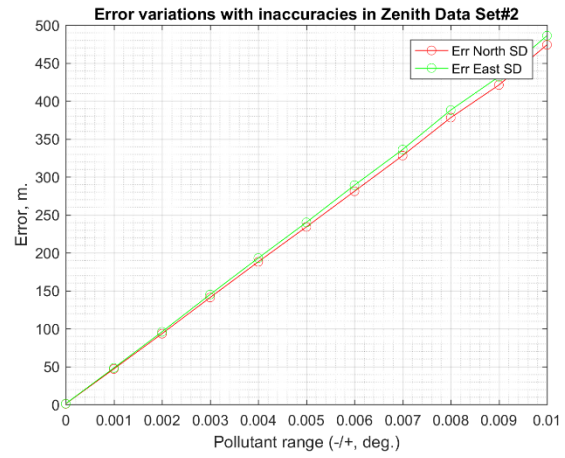
(c)

Figure C.1. Error variations with respect to inaccuracies in sensor data (a) Error variation with inaccuracies in Azimuth, (b) Error variation with inaccuracies in Zenith and (c)) Error variation with inaccuracies in Azimuth and Zenith

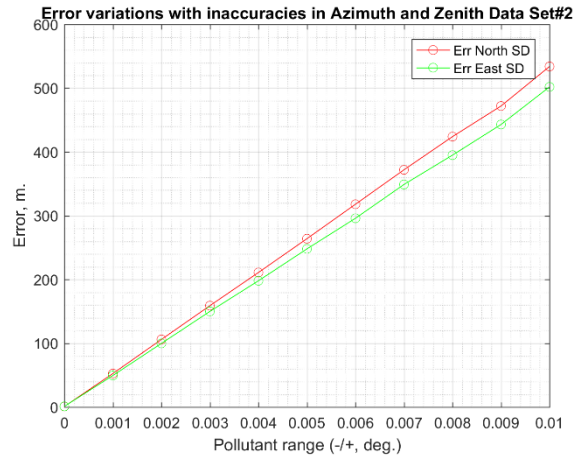
DATA SET #2



(a)



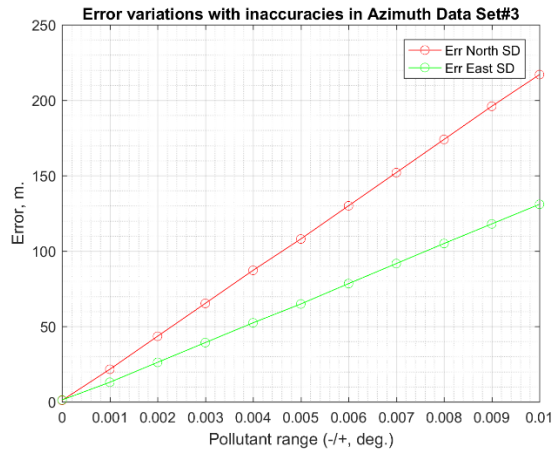
(b)



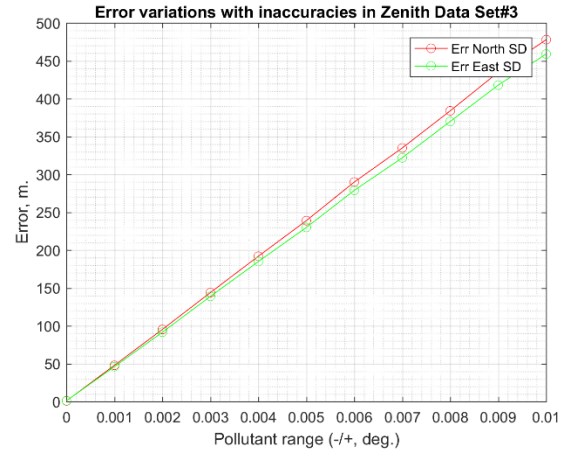
(c)

Figure C.2. Error variations with respect to inaccuracies in sensor data (a) Error variation with inaccuracies in Azimuth, (b) Error variation with inaccuracies in Zenith and (c)) Error variation with inaccuracies in Azimuth and Zenith

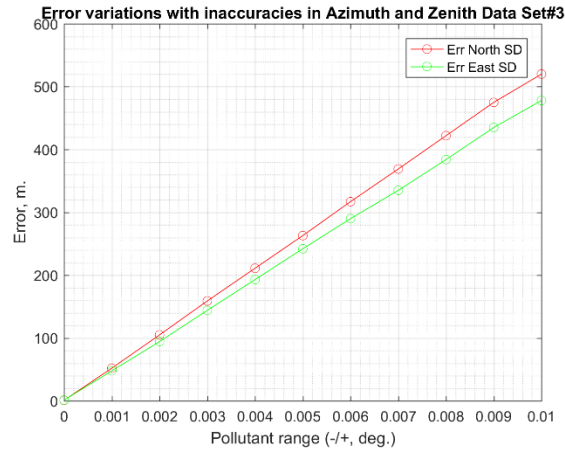
DATA SET #3



(a)



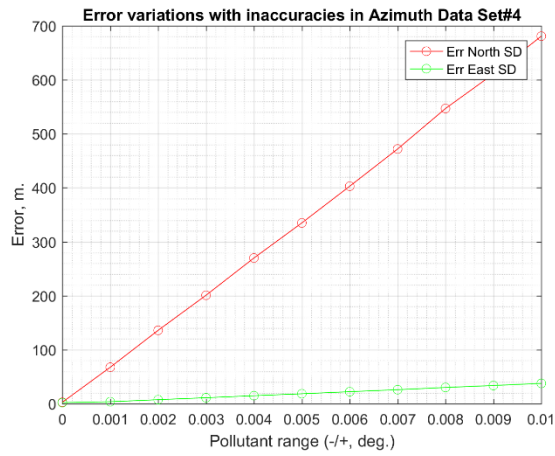
(b)



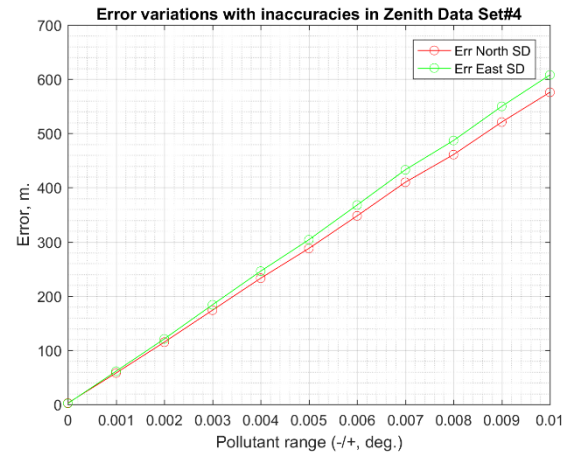
(c)

Figure C.3. Error variations with respect to inaccuracies in sensor data (a) Error variation with inaccuracies in Azimuth, (b) Error variation with inaccuracies in Zenith and (c)) Error variation with inaccuracies in Azimuth and Zenith

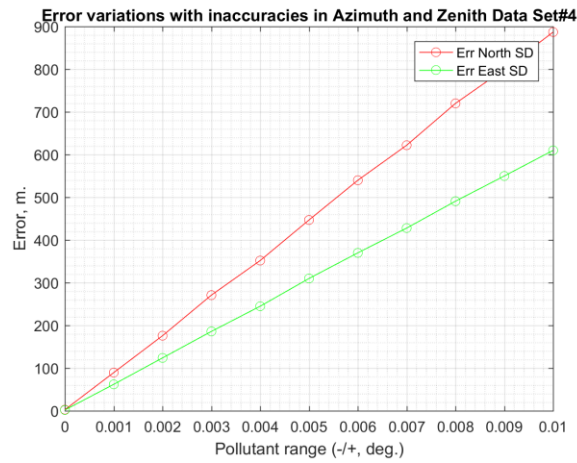
DATA SET #4



(a)



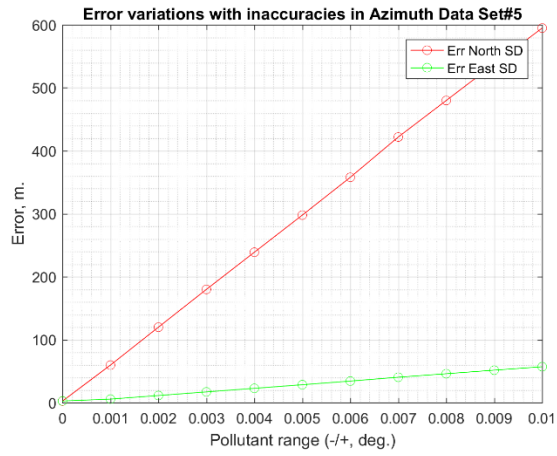
(b)



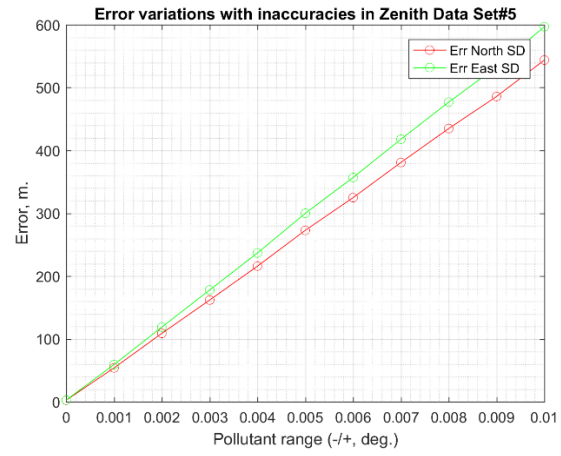
(c)

Figure C.4. Error variations with respect to inaccuracies in sensor data (a) Error variation with inaccuracies in Azimuth, (b) Error variation with inaccuracies in Zenith and (c)) Error variation with inaccuracies in Azimuth and Zenith

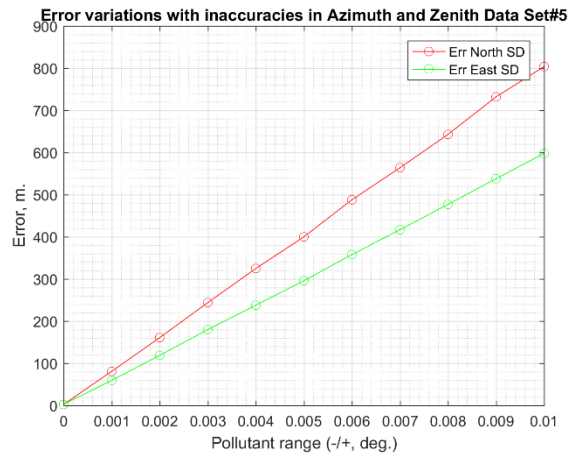
DATA SET #5



(a)



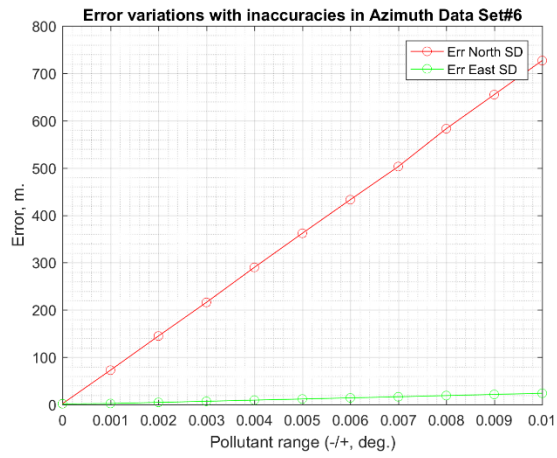
(b)



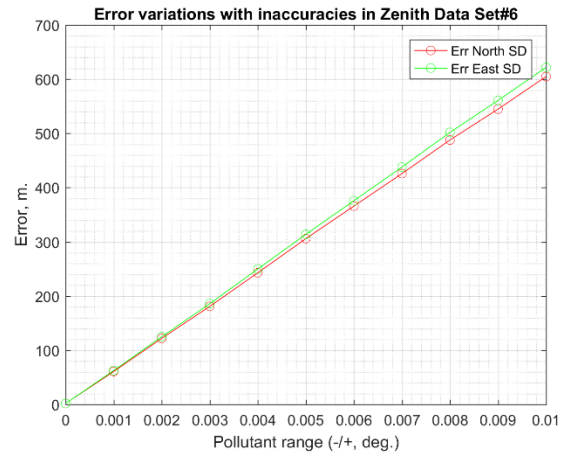
(c)

Figure C.5. Error variations with respect to inaccuracies in sensor data (a) Error variation with inaccuracies in Azimuth, (b) Error variation with inaccuracies in Zenith and (c)) Error variation with inaccuracies in Azimuth and Zenith

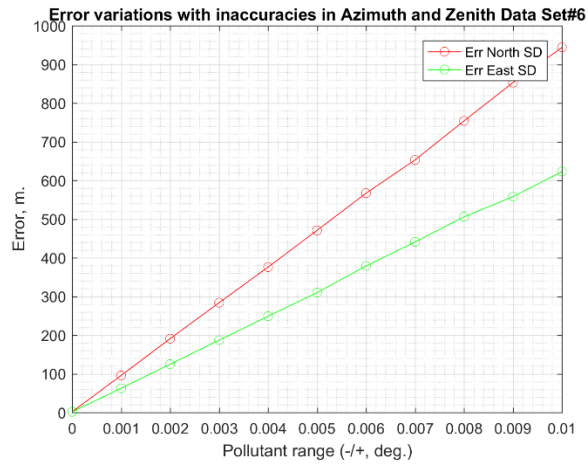
DATA SET #6



(a)



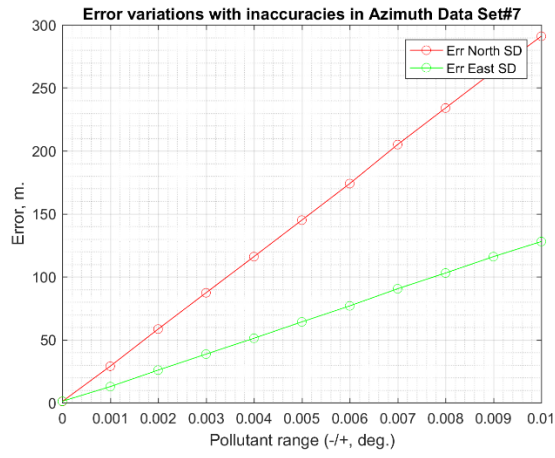
(b)



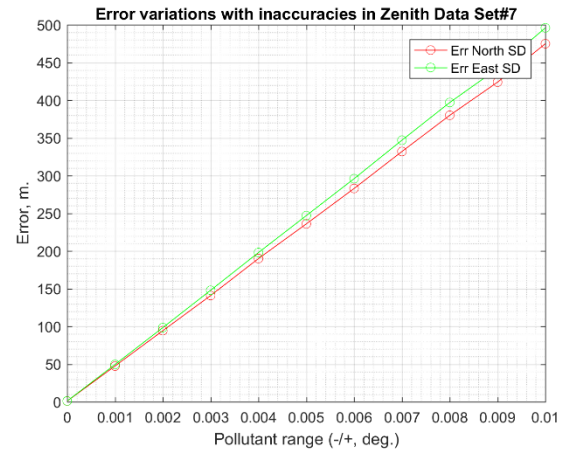
(c)

Figure C.6. Error variations with respect to inaccuracies in sensor data (a) Error variation with inaccuracies in Azimuth, (b) Error variation with inaccuracies in Zenith and (c)) Error variation with inaccuracies in Azimuth and Zenith

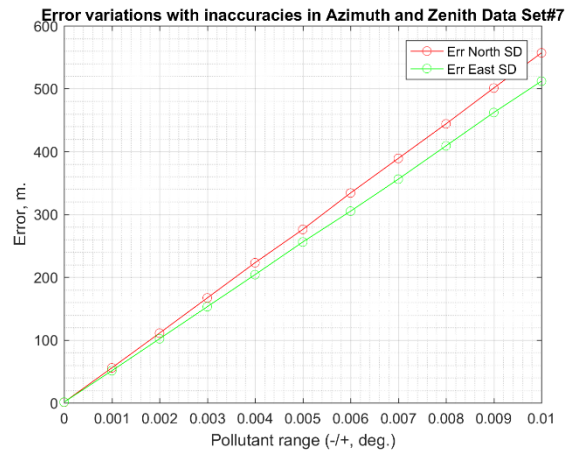
DATA SET #7



(a)



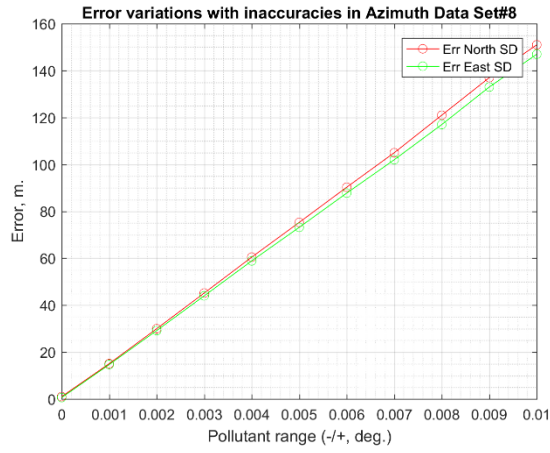
(b)



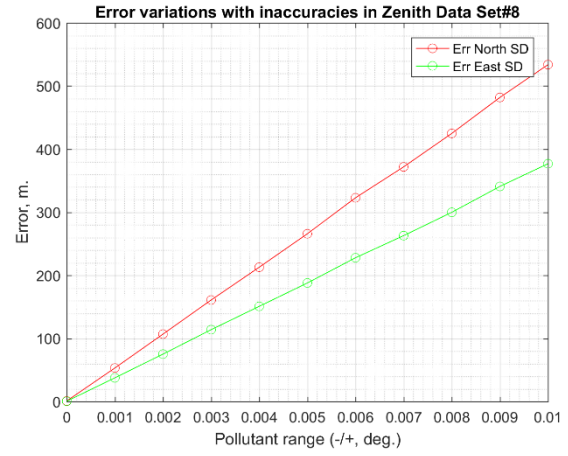
(c)

Figure C.7. Error variations with respect to inaccuracies in sensor data (a) Error variation with inaccuracies in Azimuth, (b) Error variation with inaccuracies in Zenith and (c)) Error variation with inaccuracies in Azimuth and Zenith

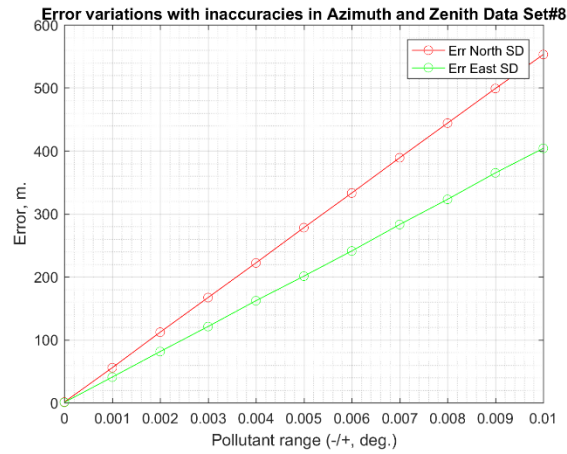
DATA SET #8



(a)



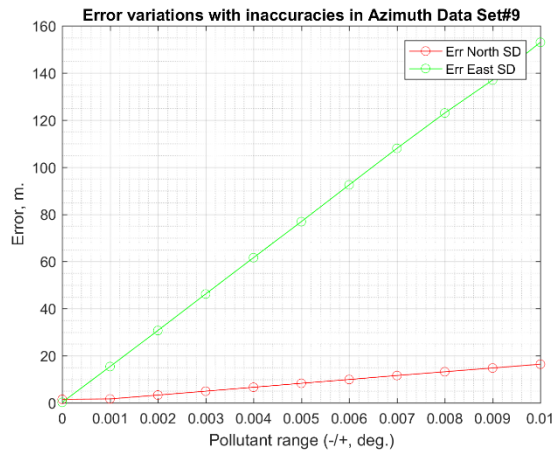
(b)



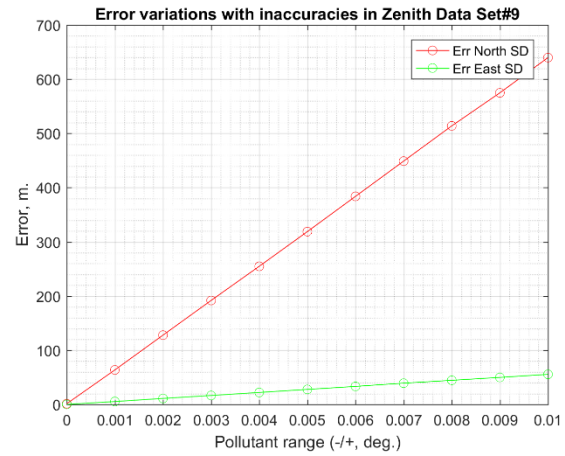
(c)

Figure C.8. Error variations with respect to inaccuracies in sensor data (a) Error variation with inaccuracies in Azimuth, (b) Error variation with inaccuracies in Zenith and (c)) Error variation with inaccuracies in Azimuth and Zenith

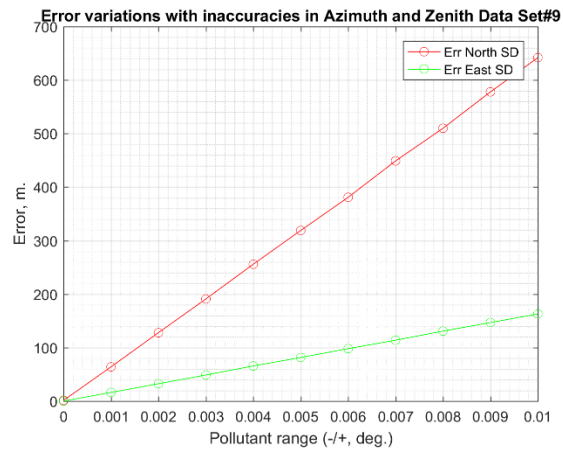
DATA SET #9



(a)



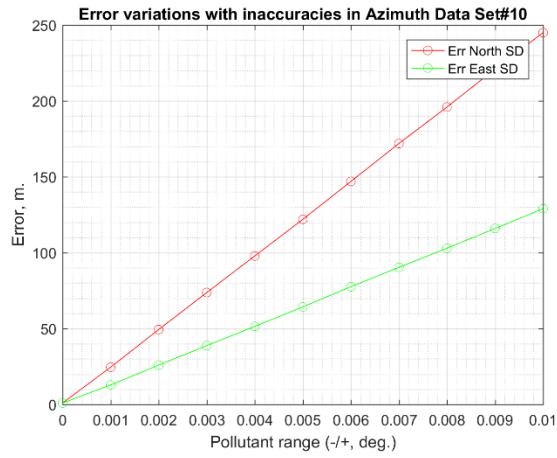
(b)



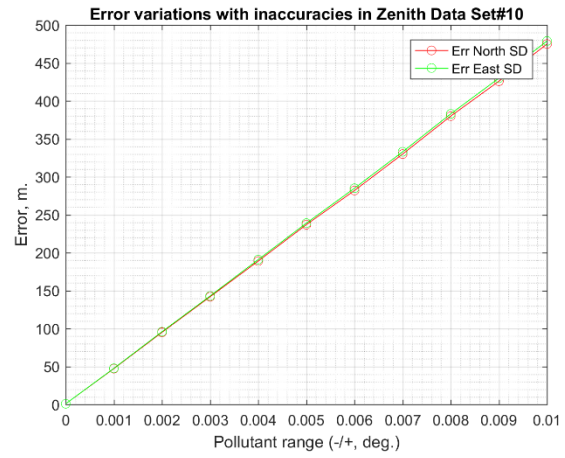
(c)

Figure C.9. Error variations with respect to inaccuracies in sensor data (a) Error variation with inaccuracies in Azimuth, (b) Error variation with inaccuracies in Zenith and (c)) Error variation with inaccuracies in Azimuth and Zenith

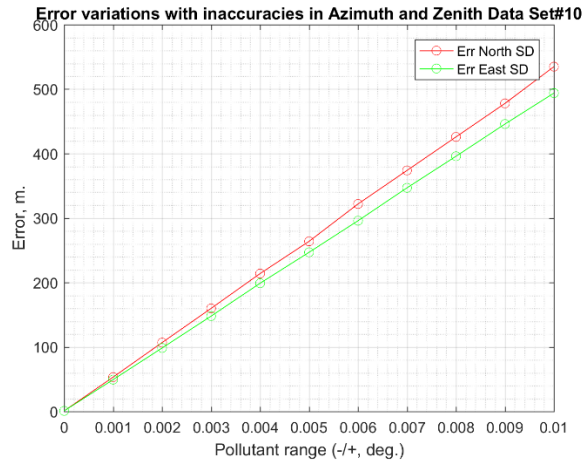
DATA SET #10



(a)



(b)



(c)

Figure C.10. Error variations with respect to inaccuracies in sensor data (a) Error variation with inaccuracies in Azimuth, (b) Error variation with inaccuracies in Zenith and (c)) Error variation with inaccuracies in Azimuth and Zenith

Appendix D. MATLAB code for ENEA algorithm

```
clc;clear all;
display('-----')
display('AE-895: MS Thesis work , Reproduction of ENEA algorithm for
computation of solar position')
display('Author: Shriniwas M Kolpuke')
display('Instructor: Dr. Shawn Keshmiri')
display('-----')

format long e
%-----
display('Inputs:')
%-----
prompt_ips = 'To Enter Longitude(deg.),Latitude(deg), P(millibars) & T(deg.
Celcius) Manually>Select::0; For Example Position(Nichols hall)>Select::1=';
ips = input(prompt_ips);

if ips == 0
    prompt_longitude = 'What is your geographical longitude(deg.)? ';
    longitude = input(prompt_longitude);
    prompt_latitude = 'What is your geographical latitude(deg.)? ';
    latitude = input(prompt_latitude);
    prompt_P = 'What is your Annual average local pressure(atm)? ';
    P = input(prompt_P);
    prompt_T = 'What is your Annual average local temperature(deg. Celcius)?
';
    T = input(prompt_T);
elseif ips == 1
    longitude = -95.264135;
    latitude = 38.952187;
    P = 0.995609365856;%atm = 1008.8011899539999; %millibars : Annual average
local pressure
    T = 12.3333; %deg._Celcius : Annual average local temperature
else
    display('Select proper input!')
    stop;
end
%-----
% display('3.1:Time calculations ENEA:')
%-----

%DAY&TIME NOW!
Day_Time = datestr(now);
display(Day_Time,'Local Date & Time - NOW!')

%UTC TIME NOW!
Time.UTC = local_time_to_utc(now);
display(Time.UTC,'UTC Date & Time - NOW!')

UTC_DateVector = datevec(Time.UTC);
%UTC_DateVector = datevec(now)
```

```

UTC_year_check = UTC_DateVector(1,1);
UTC_month_check = UTC_DateVector(1,2);
UTC_day = UTC_DateVector(1,3);
UTC_hour = UTC_DateVector(1,4);
UTC_minute = UTC_DateVector(1,5);
UTC_second = UTC_DateVector(1,6);

if UTC_month_check == 1 || UTC_month_check == 2
    UTC_month = UTC_month_check + 12;
    UTC_year = UTC_year_check - 1;
else
    UTC_month = UTC_month_check;
    UTC_year = UTC_year_check;
end

UT_hour_fraction = UTC_hour + (UTC_minute/60) + (UTC_second/(60*60));

%Delta_T for Sept.2015
delta_T = 69.4; %sec

tg = fix(365.25*(UTC_year - 2000)) + fix(30.6001*(UTC_month + 1)) + UTC_day +
(UT_hour_fraction / 24) - 1158.5;
t = tg + (delta_T/86400);

%3.2-----
% display('Calculation of the heliocentric longitude of Earth')
%-----

%Linear increasing with annual oscillation
ang = (1.72019e-2)*t - 0.0563;
Ly = 1.74094 + (1.7202768683e-2)*t + (3.34118e-2)*sin(ang) + (3.488e-
4)*sin(2*ang);

%Moon perturbation
Lm = (3.13e-5)*sin(0.2127730*t - 0.585);

%Harmonic correction
Lh = (1.26e-5)*sin((4.243e-3)*t+1.46) + (2.35e-5)*sin((1.0727e-2)*t+0.72) +
(2.76e-5)*sin((1.5799e-2)*t+2.35) + (2.75e-5)*sin((2.1551e-2)*t+1.98) +
(1.26e-5)*sin((3.1490e-2)*t+0.80);

%Polynomial correction
t2 = 0.001*t;
Lp = ( ( -(2.30796e-7)*t2 + (3.7976e-6))*t2 - (2.0458e-5) ) *t2 + (3.976e-
5) ) *t2^2;

%Longitude is sum of all these values
L = Ly + Lm + Lh + Lp;

%3.3-----
% display('Correction to geocentric longitude due to nutation')
%-----
delta_gamma = (8.33e-5)*sin((9.252e-4)*t-1.173);

```

```

%3.4-----
% display('Earth axis inclination')
%-----
epsilon = -(6.21e-9)*t+0.409086+(4.46e-5)*sin((9.252e-4)*t+0.397);

%3.5-----
% display('Geocentric global solar coordinates')
%-----

%Geocentric solar longitude
gamma = L + pi + delta_gamma - (9.932e-5);

%Geocentric right ascension
alpha = atan2( (sin(gamma)*cos(epsilon)), cos(gamma));
% % % alpha_hr = 12*mod(alpha,2*pi)/pi;

%Declination
delta = asin(sin(epsilon)*sin(gamma));

%3.6-----
% display('Local hour angle of the sun')
%-----
theta = longitude*(pi/180);
h = (6.30038809903*tg) + 4.8824623 + (0.9174*delta_gamma) + theta - alpha;

%3.7-----
% display('Parallax correction to right ascension')
%-----
psi = latitude*(pi/180);
delta_alpha = -(4.26e-5)*cos(psi)*sin(h);

%3.8-----
% display('Topocentric sun coordinates')
%-----

%Topocentric right ascension
alpha_t = alpha + delta_alpha;

%Topocentric declination
delta_t = delta - (4.26e-5)*(sin(psi) - delta*cos(psi));

%Topocentric hour angle
ht = h - delta_alpha;
cht = cos(h) + delta_alpha*sin(h);
sht = sin(h) - delta_alpha*cos(h);

%3.9-----
% display('solar elevation angle without refraction correction')
%-----
e0 = asin(sin(psi)*sin(delta_t) + cos(psi)*(cos(delta_t)) *cht);

%3.10-----
% display('Atmospheric refraction correction to the solar elevation')
%-----

```

```

e0_min = 0.01;

% delta_e = 0;
if e0 > e0_min
%     delta_e = (0.084217*P)/((273+T)/tan(e0+ (0.0031376/(e0+0.089186)))));
    delta_e = (0.084217*P)/((273+T)*tan(e0+ (0.0031376/(e0+0.089186)))));
else
    delta_e = 0;
end

%3.11-----
% display('Local topocentric sun coordinates')
%-----

%Zenith
z_zenith = ((pi/2) - e0 - delta_e);

z_zenith_deg = z_zenith*(180/pi);

%Azimuth
% Gamma_azimuth = atan2(sht, (cht*sin(psi) - tan(delta_t)*cos(psi)))
Gamma_azimuth = atan2(sht, (cht*sin(psi) - tan(delta_t)*cos(psi))) + (pi);

Gamma_azimuth_deg = Gamma_azimuth*(180/pi);

%-----
display('Sun Position:')
LastName = {'Azimuth(deg.)'; 'Zenith(deg.)'};
Angle = [Gamma_azimuth_deg; z_zenith_deg];

Outputs = table(Angle, ...
    'RowNames', LastName)
%-----

```

Appendix E. MATLAB code for SPA algorithm

```
clc;clear all;
display('-----')
display('AE-895: MS Thesis work , Reproduction of Solar Position Algorithm')
display('Author: Shriniwas M Kolpuke')
display('Instructor: Dr. Shawn Keshmiri')
display('-----')

format long e
%-----
display('Inputs:')
%-----
prompt_ips = 'To Enter Longitude(deg.),Latitude(deg), Elevation(mtrs),
P(millibars) & T(deg. Celcius) Manually>Select::0; For Example
Position(Nichols hall)>Select::1=';
ips = input(prompt_ips);

if ips == 0
    prompt_longitude = 'What is your geographical longitude(deg.)? ';
    sigma = input(prompt_longitude);
    prompt_latitude = 'What is your geographical latitude(deg.)? ';
    fi = input(prompt_latitude);
    prompt_E = 'What is your elevation(meters)? ';
    E = input(prompt_E);
    prompt_P = 'What is your Annual average local pressure(millibars)? ';
    P = input(prompt_P);
    prompt_T = 'What is your Annual average local temperature(deg. Celcius)?
';
    T = input(prompt_T);
elseif ips == 1
    sigma = -95.264135;
    fi = 38.952187;
    E = 295;
    P = 1008.8011899539999; %millibars : Annual average local pressure
    T = 12.3333; %deg._Celcius : Annual average local temperature
else
    display('Select proper input!')
    stop;
end
%3.1-----
% display('3.1:Time calculations SPA:')
%-----

%DAY&TIME NOW!
Day_Time = datestr(now);
display(Day_Time,'Local Date & Time - NOW!')

%UTC TIME NOW!
Time_UTC = local_time_to_utc(now);
display(Time_UTC,'UTC Date & Time - NOW!')

%Julian day now!
```



```

JD = juliandate(Time.UTC);

%Delta_T for Sept.2015
delta_T = 69.4; %sec

%Julian Ephemeris Day(JDE)
JDE = JD + (delta_T/86400);

%Julian Century(JC) and Julian Ephemeris Century(JCE)
JC = (JD-2451545)/36525;
JCE = (JDE-2451545)/36525;

%Julian Ephemeris Millennium(JME) for 2000 standard epoch
JME = JCE/10;

%3.2-----
% display('3.2:Calculation of the Earth heliocentric Longitude, Latitude and
Radius Vector (L, B and R):')
%-----

%-----
%Longitude(L), deg
%-----
[L0_A,L0_B,L0_C] = textread('L0.txt','%f %f %f', 64);
%length(L0_A)
L0 = 0; %initial value
for n_L0=1:1:(length(L0_A))% - 1)
    L0 = L0 + double((L0_A(n_L0,1)*double(cos( L0_B(n_L0,1) + (L0_C(n_L0,1) *
JME)))));
end

[L1_A,L1_B,L1_C] = textread('L1.txt','%f %f %f', 34);
%length(L1_A)
L1 = 0; %initial value
for n_L1=1:1:(length(L1_A))% - 1)
    L1 = L1 + (L1_A(n_L1,1)*cos( L1_B(n_L1,1) + (L1_C(n_L1,1) * JME)));
end

[L2_A,L2_B,L2_C] = textread('L2.txt','%f %f %f', 20);
% length(L2_A)
L2 = 0; %initial value
for n_L2=1:1:(length(L2_A))% - 1)
    L2 = L2 + (L2_A(n_L2,1)*cos( L2_B(n_L2,1) + (L2_C(n_L2,1) * JME)));
end

[L3_A,L3_B,L3_C] = textread('L3.txt','%f %f %f', 7);
% length(L3_A)
L3 = 0; %initial value
for n_L3=1:1:(length(L3_A))% - 1)
    L3 = L3 + (L3_A(n_L3,1)*cos( L3_B(n_L3,1) + (L3_C(n_L3,1) * JME)));
end

[L4_A,L4_B,L4_C] = textread('L4.txt','%f %f %f', 3);
% length(L4_A)
L4 = 0; %initial value

```

```

for n_L4=1:1:(length(L4_A))% - 1)
    L4 = L4 + (L4_A(n_L4,1)*cos( L4_B(n_L4,1) + (L4_C(n_L4,1) * JME)));
end

[L5_A,L5_B,L5_C] = textread('L5.txt','%f %f %f', 1);
% length(L5_A)
L5 = 0; %initial value
for n_L5=1:1:(length(L5_A))% - 1)
    L5 = L5 + (L5_A(n_L5,1)*cos( L5_B(n_L5,1) + (L5_C(n_L5,1) * JME)));
end

L_rad = (L0 + (L1*JME) + (L2*(JME)^2) + (L3*(JME)^3) + (L4*(JME)^4) +
(L5*(JME)^5))/(10^8);

L_deg = (L_rad * 180)/pi;

F_L = abs(rem((L_deg/360),1));

if L_deg > 0
    L = 360 * F_L;
elseif L_deg < 0
    L = 360 - (360*F_L);
end
% display(L,'Longitude(L), deg')

%-----
%Latitude(B), deg
%-----
[B0_A,B0_B,B0_C] = textread('B0.txt','%f %f %f', 5);
% length(B0_A)
B0 = 0; %initial value
for n_B0=1:1:(length(B0_A))% - 1)
    B0 = B0 + (B0_A(n_B0,1)*cos( B0_B(n_B0,1) + (B0_C(n_B0,1) * JME)));
end

[B1_A,B1_B,B1_C] = textread('B1.txt','%f %f %f', 2);
% length(B1_A)
B1 = 0; %initial value
for n_B1=1:1:(length(B1_A))% - 1)
    B1 = B1 + (B1_A(n_B1,1)*cos( B1_B(n_B1,1) + (B1_C(n_B1,1) * JME)));
end

B2 = 0;
B3 = 0;
B4 = 0;
B5 = 0;

B_rad = (B0 + (B1*JME) + (B2*(JME)^2) + (B3*(JME)^3) + (B4*(JME)^4) +
(B5*(JME)^5))/(10^8);

B = (B_rad * 180)/pi;

% display(B,'Latitude(B), deg')

```

```

%-----
%Earth radius vector(R), AU
%-----
[R0_A,R0_B,R0_C] = textread('R0.txt','%f %f %f', 64);
% length(R0_A)
R0 = 0; %initial value
for n_R0=1:1:(length(R0_A))% - 1)
    R0 = R0 + (R0_A(n_R0,1)*cos( R0_B(n_R0,1) + (R0_C(n_R0,1) * JME)));
end

[R1_A,R1_B,R1_C] = textread('R1.txt','%f %f %f', 10);
% length(R1_A)
R1 = 0; %initial value
for n_R1=1:1:(length(R1_A))% - 1)
    R1 = R1 + (R1_A(n_R1,1)*cos( R1_B(n_R1,1) + (R1_C(n_R1,1) * JME)));
end

[R2_A,R2_B,R2_C] = textread('R2.txt','%f %f %f', 6);
% length(R2_A)
R2 = 0; %initial value
for n_R2=1:1:(length(R2_A))% - 1)
    R2 = R2 + (R2_A(n_R2,1)*cos( R2_B(n_R2,1) + (R2_C(n_R2,1) * JME)));
end

[R3_A,R3_B,R3_C] = textread('R3.txt','%f %f %f', 2);
% length(R3_A)
R3 = 0; %initial value
for n_R3=1:1:(length(R3_A))% - 1)
    R3 = R3 + (R3_A(n_R3,1)*cos( R3_B(n_R3,1) + (R3_C(n_R3,1) * JME)));
end

[R4_A,R4_B,R4_C] = textread('R4.txt','%f %f %f', 1);
% length(R4_A)
R4 = 0; %initial value
for n_R4=1:1:(length(R4_A))% - 1)
    R4 = R4 + (R4_A(n_R4,1)*cos( R4_B(n_R4,1) + (R4_C(n_R4,1) * JME)));
end

R5 = 0;

R_AU = (R0 + (R1*JME) + (R2*(JME)^2) + (R3*(JME)^3) + (R4*(JME)^4) +
(R5*(JME)^5))/(10^8);
% display(R_AU,'Earth radius vector(R), AU')

%3.3-----
% display('3.3:Calculation of the geocentric Longitude(theta) and
Latitude(beta):')
%-----

%-----
%Longitude(theta), deg
%-----
theta_initial = L + 180;
F_theta = abs(rem((theta_initial/360),1));

```

```

if theta_initial > 0
    theta = 360 * F_theta;
elseif theta_initial < 0
    theta = 360 - (360 * F_theta);
end
% display(theta, 'Longitude(theta), deg')

%-----
%Latitude(beta), deg
%-----
beta = -B;
% display(beta, 'Latitude(beta), deg')

%3.4-----
% display('3.4:Calculation of the nutation in Longitude(delta_si) and
Obliquity(delta_epsilon):')
%-----

%Calculation of mean elongation of the moon from the sun: X0, deg.
X0 = 297.85036 + 445267.111480 * JCE - 0.0019142 * (JCE)^2 + (((JCE)^3)/189474);
% display(X0, 'Mean elongation of the moon from the sun: X0, deg')

%Calculation of mean anomaly of the sun(Earth): X1, deg.
X1 = 357.52772 + 35999.050340 * JCE - 0.0001603 * (JCE)^2 - (((JCE)^3)/300000);
% display(X1, 'Mean anomaly of the sun(Earth): X1, deg')

%Calculation of mean anomaly of the moon: X2, deg.
X2 = 134.96298 + 477198.867398 * JCE + 0.0086972 * (JCE)^2 + (((JCE)^3)/56250);
% display(X2, 'Mean anomaly of the moon: X2, deg')

%Calculation of the moon's argument of latitude: X3, deg.
X3 = 93.27191 + 483202.017538 * JCE - 0.0036825 * (JCE)^2 + (((JCE)^3)/327270);
% display(X3, 'Moon"s argument of latitude: X3, deg')

%Calculation of the longitude of the ascending node of the moon's mean
%orbit on the ecliptic, measured from the mean equinox of the date: X4,
%deg.
X4 = 125.04452 - 1934.136261 * JCE + 0.0020708 * (JCE)^2 + (((JCE)^3)/450000);
% display(X4, 'Longitude of the ascending node of the moon"s mean orbit on the
ecliptic, measured from the mean equinox of the date: X4, deg.')

X_j = [X0; X1; X2; X3; X4];

%Calculation of delta_si_i and delta_epsilon_i (in 0.0001 of arc seconds)
[a_i, b_i, c_i, d_i] = textread('PE_TERMS.txt', '%f %f %f %f', 63);
[Y0, Y1, Y2, Y3, Y4] = textread('Y_TERMS.txt', '%f %f %f %f %f', 63);
Y_ij = [Y0, Y1, Y2, Y3, Y4];
Y = 0;
X = 0;
for j=1:1:5
    for i=1:1:63
        %Y_j= +y_i;
        Y = Y + Y_ij(i,j);
    end
    %x_j= +x_j;

```

```

        X = X + X_j(j);
end
xy_sin =sin(X*Y);

A_i = 0;
B_i = 0;
for i=1:1:63
    A_i = A_i + a_i(i);
    B_i = B_i + b_i(i);
end
B_JCE = B_i * JCE;
AB_JCE = A_i + B_JCE;
delta_si_i = AB_JCE * xy_sin;
delta_si = delta_si_i/36000000;

C_i = 0;
D_i = 0;
xy_cos =cos(X*Y);
D_JCE = D_i * JCE;
CD_JCE = C_i + D_JCE;
delta_epsilon_i = CD_JCE * xy_cos;
delta_epsilon = delta_epsilon_i/36000000;
for i=1:1:63
    C_i = C_i + c_i(i);
    D_i = D_i + d_i(i);
end

%3.5-----
% display('3.5:Calculation of the true obliquity of the ecliptic(epsilon,
deg.):')
%-----

%Calculation of the mean obliquity of the ecliptic, epsilon0(in arc seconds)
U = JME/10;
epsilon0 = 84381.448 - 4680.93*(U) - 1.55*(U^2) + 1999.25*(U^3) - 51.38*(U^4)
- 249.67*(U^5) - 39.05*(U^6) + 7.12*(U^7) + 27.87*(U^8) + 5.79*(U^9) +
2.45*(U^10);

%Calculation of the true obliquity of the ecliptic, (epsilon, deg.)
epsilon = (epsilon0/3600) + delta_epsilon;
% display(epsilon,'epsilon, deg.')

%3.6-----
% display('3.6:Calculation of the aberration correction, (delta_Tau, deg.):')
%-----
delta_Tau = - double(20.4898 / double(3600*R_AU));
% display(delta_Tau,'delta_Tau, deg.')

%3.7-----
% display('3.7:Calculation of the apparent sun longitude, (lambda, deg.):')
%-----
lambda = theta+delta_si+delta_Tau;
% display(lambda,'lambda, deg.')

%3.8-----

```

```

% display('3.8:Calculation of the apparent sidereal time at Greenwich at any
given time, (V, deg):')
%-----

%Calculation of the mean sidereal time at Greenwich, (V0, deg.)
V0_i = 280.46061837 + 360.98564736629*(JD-2451545) + 0.000387933*(JC^2) -
((JC^3)/38710000);
F_V0 = abs(rem((V0_i/360),1));

if V0_i > 0
    V0 = 360 * F_V0;
elseif V0_i < 0
    V0 = 360 - (360*F_V0);
end
% display(V0,'V0, deg')

%Calculation of the apparent sidereal time at Greenwich, (V, deg.)
V = V0 + (delta_si * cos(epsilon));
% display(V,'V, deg')

%3.9-----
% display('3.9:Calculation of the geocentric sun right ascension, (alfa,
deg):')
%-----
alfa_rad = atan2 ( ((sin(lambda*(pi/180))*cos(epsilon*(pi/180)))-
(tan(beta*(pi/180))*sin(epsilon*(pi/180)))), cos(lambda*(pi/180)) );

alfa_deg = (alfa_rad * 180)/pi;

F_alfa_deg = abs(rem((alfa_deg/360),1));

if alfa_deg > 0
    alfa = 360 * F_alfa_deg;
elseif alfa_deg < 0
    alfa = 360 - (360*F_alfa_deg);
end
% display(alfa,'alfa, deg')

%3.10-----
% display('3.10:Calculation of the geocentric sun declination, (delta,
deg):')
%-----
delta_rad = asin((sin(beta*(pi/180))*cos(epsilon*(pi/180))) +
(cos(beta*(pi/180))*sin(epsilon*(pi/180))*sin(lambda*(pi/180)));
delta = (delta_rad * 180)/pi;
% display(delta,'delta, deg.')

%3.11-----
% display('3.11:Calculation of the observer local hour angle, (H, deg):')
%-----
H_i = V + sigma - alfa;

F_H_i = abs(rem((H_i/360),1));

```

```

if H_i > 0
    H = 360 * F_H_i;
elseif H_i < 0
    H = 360 - (360 * F_H_i);
end
% display(H, 'H, deg.')

%3.12-----
% display('3.12:Calculation of the topocentric sun right ascension,
(alfa_dash, deg.):')
%-----
%Topocentric means that the sun position is calculated wrt the observer
%local position at the Earth surface.

%Calculation of the equatorial horizontal parallax of the sun, (Xi, deg.)
Xi = (8.794)/(3600*R_AU);
% display(Xi, 'Xi, deg.')

%Calculation the term u(rad)
u = atan(0.99664719*tan(fi*(pi/180))); %0.99664719=(1-f); f-Earth's
flattening
% display(u, 'u, rad.')

%Calculation of term x
x = cos(u) + ((E/6378140) * cos(fi*(pi/180)));
% display(x, 'x')
%x=ro_dash*cos(fi_dash); fi_dash-observer's geocentric latitude,
%ro-observer's distance to the center of Earth

%Calculation of term y
y = (0.99664719*sin(u)) + ((E/6378140)*sin(fi*(pi/180)));
% display(y, 'y')
%y=ro_dash*sin(fi_dash)

%Calculation of the parallax in the sun right ascension, (delta_alfa, deg.)
delta_alfa_rad = atan2((-
x*sin(Xi*(pi/180))*sin(H*(pi/180))), (cos(delta*(pi/180))-
(x*sin(Xi*(pi/180))*cos(H*(pi/180))));
delta_alfa = (delta_alfa_rad * 180)/pi;
% display(delta_alfa, 'delta_alfa, deg.')

%Calculation of the topocentric sun right ascension, (alfa_dash, deg.)
alfa_dash = alfa + delta_alfa;
% display(alfa_dash, 'alfa_dash, deg.')

%Calculation of the topocentric sun declination, (delta_dash, deg.)
delta_dash_rad = atan2(((sin(delta*(pi/180)) -
(y*sin(Xi*(pi/180))))*cos(delta_alfa*(pi/180))), (cos(delta*(pi/180))-
(x*sin(Xi*(pi/180))*cos(H*(pi/180))));
delta_dash = delta_dash_rad * (180/pi);
% display(delta_dash, 'delta_dash, deg.')

%3.13-----
% display('3.13:Calculation of the topocentric local hour angle, (H_dash,
deg.):')

```

```

%-----
H_dash = H - delta_alfa;
% display(H_dash,'H_dash, deg.')

%3.14-----
% display('3.14:Calculation of the topocentric zenith angle, (theta_zenith,
deg):')
%-----

%Calculation of the topocentric elevation angle without atmospheric
%refraction correction, (e0, deg)
e0_rad = asin( (sin(fi*(pi/180))*sin(delta*(pi/180))) +
(cos(fi*(pi/180))*cos(delta*(pi/180))*cos(H_dash*(pi/180))) );
e0 = (e0_rad * 180)/pi;
% display(e0,'e0, deg')

%Calculation of the atmospheric refraction correction, (delta_e, deg)
delta_e = (P/1010)*(283/(273+T))*(1.02/(60*tan(e0+(10.3/(e0+5.11)))));
% display(delta_e,'delta_e, deg.')

%Calculation of topocentric elevation angle, (e, deg)
e = e0 + delta_e;
% display(e,'e, deg.')

%Calculation of the topocentric zenith angle, (theta_zenith, deg)
theta_zenith = 90 - e;
% display(theta_zenith,'theta_zenith, deg.')

%3.15-----
% display('3.15:Calculation of the topocentric azimuth angle, (phi, deg):')
%-----

%Calculation of the topocentric astronomers azimuth angle, (Gamma, deg.)
Gamma_rad = atan2
((sin(H_dash*(pi/180))), ((cos(H_dash*(pi/180))*sin(fi*(pi/180)))-
(tan(delta_dash*(pi/180))*cos(fi*(pi/180))));
Gamma_deg = (Gamma_rad * 180)/pi;

F_Gamma_deg = abs(rem((Gamma_deg/360),1));

if Gamma_deg > 0
    Gamma = 360 * F_Gamma_deg;
elseif Gamma_deg < 0
    Gamma = 360 - (360*F_Gamma_deg);
end
% display(Gamma,'Gamma, deg')

%Calculation of topocentric azimuth angle, phi for navigators and solar
%radiation users, (phi, deg.)
phi_deg = Gamma + 180;

F_phi_deg = abs(rem((phi_deg/360),1));

if phi_deg > 0

```



```

    phi = 360 * F_phi_deg;
elseif phi_deg < 0
    phi = 360 - (360 * F_phi_deg);
end
% display(phi, 'phi, deg')

%3.16-----
% display('3.16:Calculation of the incidence angle for a surface oriented in
any direction, (I, deg):')
%-----
omega = 0; %Slope of surface measured from horizontal plane
gama = 0; %Surface azimuth rotation angle, measured from south to the
projection of the surface normal on the horizontal plane, positive or
negative if oriented west or east from south respectively.

I_rad =
acos((cos(theta_zenith*(pi/180))*cos(omega*(pi/180)))+(sin(omega*(pi/180))*si
n(theta_zenith*(pi/180))*cos(Gamma*(pi/180) - gama*(pi/180)));
I = I_rad * (180/pi);
% display(I, 'I, deg.')

%*****
% display('-----')
% % display(Day_Time, 'Local Date & Time - NOW!')
% % display(Time_UTC, 'UTC Date & Time - NOW!')
% % display(phi, 'Top. azimuth angle (eastward from N), deg.')
% % display(I, 'Surface incidence angle, deg.')
% display('-----')
%-----
display('Sun Position:')
LastName = {'Azimuth(deg.)'; 'Incidence(deg.)'};
Angle = [phi; I];

Outputs = table(Angle, ...
    'RowNames', LastName)
%-----

```

Appendix F. MATLAB code for Michalsky algorithm

```
clc;clear all;
display('-----')
display('AE-895: MS Thesis work , Reproduction of Michalsky algorithm for
computation of solar position')
display('Author: Shriniwas M Kolpuke')
display('Instructor: Dr. Shawn Keshmiri')
display('-----')

format long e
%-----
display('Inputs:')
%-----
prompt_ips = 'To Enter Longitude & Latitude (deg) Manually>Select::0; For
Example Position(Nichols hall)>Select::1=';
ips = input(prompt_ips);

if ips == 0
    prompt_longitude = 'What is your geographical longitude(deg.)? ';
    longitude = input(prompt_longitude);
    prompt_latitude = 'What is your geographical latitude(deg.)? ';
    latitude = input(prompt_latitude);
elseif ips == 1
    longitude = -95.264135;
    latitude = 38.952187;
else
    display('Select proper input!')
    stop;
end

%-----
% display('1:Time calculations Michalsky:')
%-----

%DAY&TIME NOW!
Day_Time = datestr(now);
display(Day_Time,'Local Date & Time - NOW!')

%UTC TIME NOW!
Time_UTC = local_time_to_utc(now);
display(Time_UTC,'UTC Date & Time - NOW!')

UTC_DateVector = datevec(Time_UTC);

UTC_year = UTC_DateVector(1,1);
UTC_hour = UTC_DateVector(1,4);
UTC_minute = UTC_DateVector(1,5);
UTC_second = UTC_DateVector(1,6);

UTC_hour_fraction = UTC_hour + (UTC_minute/60) + (UTC_second/(60*60));
```

```

v = datenum(UTC_DateVector);
year_day = fix(v - datenum(UTC_DateVector(1), 1,0)); % datenum(yr,1,0) ==
datenum(yr-1,12,31)

delta = UTC_year -1949;
leap = fix(delta/4);

%Julian day now!
%JD = juliandate(Time.UTC);
JD = 2432916.5 + (delta*365) + leap + year_day + (UTC_hour_fraction/24);

%The input for the astronomer's almanach is the difference between the
%julian date and JD 2451545.0 (noon, 1 January 2000)
time = JD - 2451545;%51545

%-----
%display('2:Anomaly and obliquity calculations:')
%-----

%Mean longitude
mean_long_deg = 280.460 + (0.9856474 * time);

F_mean_long_deg = abs(rem((mean_long_deg/360),1));
if mean_long_deg > 0
    mean_long = 360 * F_mean_long_deg;
elseif mean_long_deg <0
    mean_long = 360 - (360*F_mean_long_deg);
end

%Mean anomaly
mean_anom_deg = 357.528 + (0.9856003 * time);

F_mean_anom_deg = abs(rem((mean_anom_deg/360),1));
if mean_anom_deg > 0
    mean_anom = 360 * F_mean_anom_deg;
elseif mean_anom_deg <0
    mean_anom = 360 - (360*F_mean_anom_deg);
end

mean_anom_rad = mean_anom * (pi/180);

%Ecliptic longitude and obliquity of ecliptic
ecliptic_long_deg = (mean_long_deg + (1.915*sin(mean_anom_rad)) +
(0.020*sin(2*mean_anom_rad)));

F_ecliptic_long_deg = abs(rem((ecliptic_long_deg/360),1));
if ecliptic_long_deg > 0
    ecliptic_long = 360 * F_ecliptic_long_deg;
elseif ecliptic_long_deg <0
    ecliptic_long = 360 - (360*F_ecliptic_long_deg);
end

ecliptic_long_rad = ecliptic_long * (pi/180);

```

```

ecliptic_obliquity_rad = (23.439 - (0.0000004*time))*(pi/180);
%-----
%display('3:Celestial coordinates:')
%-----

%Right ascension and declination
num = cos(ecliptic_obliquity_rad)*sin(ecliptic_long_rad);
den = cos(ecliptic_long_rad);

right_ascension_rad_check = atan(num/den);

if den < 0
    right_ascension_rad = right_ascension_rad_check + pi;
elseif num < 0
    right_ascension_rad = right_ascension_rad_check + (2*pi);
end

declination_rad = asin(sin(ecliptic_obliquity_rad) * sin(ecliptic_long_rad));

%-----
%display('4:Local coordinates:')
%-----

%Greenwith mean sidereal time
gmst_check = (6.697375 + (0.0657098242*time) + UTC_hour_fraction);

F_gmst_check = abs(rem((gmst_check/24),1));
if gmst_check > 0
    gmst = 24 * F_gmst_check;
elseif gmst_check <0
    gmst = 24 - (24*F_gmst_check);
end

%Local mean sidereal time
lmst_check = (gmst + ( longitude/15));

F_lmst_check = abs(rem((lmst_check/24),1));
if lmst_check > 0
    lmst = 24 * F_lmst_check;
elseif lmst_check <0
    lmst = 24 - (24*F_lmst_check);
end

lmst_rad = (lmst * 15) * (pi/180);

%Hour angle (rad)
hour_angle_rad_check = (lmst_rad - right_ascension_rad);

F_hour_angle_rad_check = abs(rem((hour_angle_rad_check/(2*pi)),1));
if hour_angle_rad_check > 0
    hour_angle_rad = (2*pi) * F_hour_angle_rad_check;
elseif hour_angle_rad_check <0
    hour_angle_rad = (2*pi) - ((2*pi)*F_hour_angle_rad_check);
end

```

```

%-----
%display('5:Elevation and azimuth:')
%-----
%Elevation
latitude_rad = latitude * (pi/180);
elevation_rad = asin(sin(declination_rad)*sin(latitude_rad) +
cos(declination_rad)*cos(latitude_rad)*cos(hour_angle_rad));

%Azimuth
azimuth_rad_check = asin(-
cos(declination_rad)*(sin(hour_angle_rad)/cos(elevation_rad)));

if (sin(declination_rad) - sin(elevation_rad)*sin(latitude_rad)) <0
    azimuth_rad = pi - azimuth_rad_check;
elseif (azimuth_rad_check) < 0
    azimuth_rad = azimuth_rad_check + (2*pi);
end

%Print
elevation = elevation_rad * (180/pi);
azimuth = azimuth_rad * (180/pi);

%-----
display('Sun Position:')
LastName = {'Azimuth(deg.)'; 'Elevation(deg.)'};
Angle = [azimuth;elevation];

Outputs = table(Angle,...
    'RowNames',LastName)
%-----

```